**Реферат**

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Рыжанкова А.С.*

*Вольский Д.М.*

*Жиляк Н.А.*

Ф.И.О.

Подпись

Дата

1

*Реферат*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*74218005, 2020*

Пояснительная записка дипломного проекта содержит 72 страницы, 56 рисунков, 13 источников литературы, 5 приложений.

ВЕБ-ПРИЛОЖЕНИЕ, ASP.NET CORE, XAMARIN.FORMS, C#, MVC, ENTITY FRAMEWORK CORE, SQL, JAVASCRIPT, JQUERY

Целью дипломного проекта является разработка веб-приложения для генерации интреактивных интерфейсов под операционную систему Android.

В первой главе проводится аналитический обзор литературы по тематике дипломного проекта.

Вторая глава посвящена обзору средств разработки и содержит описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвертой главе описывается контрольный пример, с проведением тестирования, а также демонстрация поведения системы в различных внештатных ситуациях.

В пятой главе описано руководство пользователя, позволяющее подробно понять интерфейс программного средства.

В шестой главе приводится расчет экономических параметров и себестоимости программного средства, разработанного в рамках дипломного проекта.

Объем графической части дипломного проекта составляет 1,25 листа А1.

*Жиляк Н.А.*

Консульт.

**Abstract**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Вольский Д.М.*

Пров.

Жиляк Н.А.

Консульт.

Жиляк Н.А.

Н. контр.

Рыжанкова А.С.

Утв.

Смелов В.В.

Abstruct

Лит.

Листов

1

*74218005, 2020*

The explanatory note contains 72 pages, 56 illustrations, 13 literature sources, 5 appendices.

WEB-APPLICATION, ASP.NET CORE, XAMARIN.FORMS, C#, MVC, ENTITY FRAMEWORK CORE, SQL, JAVASCRIPT, JQUERY

The aim of the thesis project is to develop a web application for generating interactive interfaces for the Android operating system. The explanatory note of the diploma project consists of an abstract, table of contents, six chapters, conclusion and list of sources used.

In the first section, the tasks of software development are formulated, an overview of analogues with the pros and cons of each of them, as well as a brief theoretical information on the topic of work.

The second section contains a description of the technologies selected for the project, as well as a description of the architecture of the software being developed.

The third section contains a phased description of the project implementation, including code listings with comments explaining the reason for choosing one or another implementation.

The fourth section describes the system testing process in the case of various user scenarios.

The fifth section describes the installation and use of the user manual with a description of the functionality of the system.

The sixth section includes the calculation of the cost and selling price of the developed software.

In conclusion, the results of software development are listed, listing the tasks performed during the work on the project.

The volume of the graphic part of the diploma project is 1.25 sheets A1.

**Содержание**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Вольский Д.М.*

Пров.

Жиляк Н.А.

Консульт.

Жиляк Н.А.

Н. контр.

Рыжанкова А.С.

Утв.

Смелов В.В.

Содержание

Лит.

Листов

2

*74218005, 2020*

[Введение 7](#_Toc41983549)

[1 Обзор теоретического материала и аналогов 8](#_Toc41983550)

[1.1 Основные сведения по теме «Прототипирования» 8](#_Toc41983551)

[1.2 Типы и классификации прототипов 9](#_Toc41983552)

[1.3 Обзор аналогов 9](#_Toc41983553)

[1.3.1 Сервис создания динамических интерфейсов «Marvelapp» 10](#_Toc41983554)

[1.3.2 Сервис создания динамических интерфейсов «InVision» 11](#_Toc41983555)

[1.3.3 Сервис создания динамических интерфейсов «Origami Studio» 12](#_Toc41983556)

[1.4 Патентный поиск по теме дипломного проекта 13](#_Toc41983557)

[1.4.1 Веб-приложение-генератор для мобильных устройств 14](#_Toc41983558)

[1.5 Выводы по разделу 14](#_Toc41983559)

[2 Технические средства разработки 16](#_Toc41983560)

[2.1 Описание работы и структуры программного средства 17](#_Toc41983561)

[2.2 Обоснование выбора языка программирования 18](#_Toc41983562)

[2.3 Обзор средств разработки 19](#_Toc41983563)

[2.3.1 Платформа .NET 19](#_Toc41983564)

[2.3.2 Платформа ASP.NET Core 19](#_Toc41983565)

[2.3.3 Платформа ASP.NET Core MVC 20](#_Toc41983566)

[2.3.4 База данных PostgreSQL 21](#_Toc41983567)

[2.3.5 Структурированный язык запросов SQL 22](#_Toc41983568)

[2.3.6 Технология Entity Framework Сore 22](#_Toc41983569)

[2.3.7 Язык программирования JavaScript 24](#_Toc41983570)

[2.3.8 Библеотека jQuery 24](#_Toc41983571)

[2.3.9 Платформа Xamarin.Forms 25](#_Toc41983572)

[2.4 Выводы по разделу 26](#_Toc41983573)

[3 Разработка программного обеспечения веб-приложения 27](#_Toc41983574)

[3.1 Проектирование программного средства 27](#_Toc41983575)

[3.2 Файловая структура 28](#_Toc41983576)

[3.3 Структура базы данных 29](#_Toc41983577)

[3.4 Подключение к базе данных 32](#_Toc41983578)

[3.5 Страница авторизации 34](#_Toc41983579)

[3.6 Страница списка проектов 36](#_Toc41983580)

[3.7 Страница редактирования проекта 38](#_Toc41983581)

[3.8 Режим предпросмотра 47](#_Toc41983582)

[3.9 Генератор мобильного приложения 47](#_Toc41983583)

[3.10 Вывод по разделу 51](#_Toc41983584)

[4 Тестирование программного средства 53](#_Toc41983585)

[4.1 Тестирование продукта с помощью unit тестов 53](#_Toc41983586)

[4.2 Тестирование создания пользователя 54](#_Toc41983587)

[4.3 Тестирование создания проекта 55](#_Toc41983588)

[4.4 Тестирование удаления проекта 55](#_Toc41983589)

[4.5 Тестирование редактирования проекта 56](#_Toc41983590)

[4.6 Выводы по разделу 57](#_Toc41983591)

[5 Руководство пользователя 58](#_Toc41983592)

[5.1 Авторизация пользователя 58](#_Toc41983593)

[5.2 Регистрация пользователя 59](#_Toc41983594)

[5.3 Страница списка проектов 60](#_Toc41983595)

[5.4 Страница редактирования проекта 61](#_Toc41983596)

[5.5 Генерация мобильного приложения 63](#_Toc41983597)

[5.6 Выводы по разделу 64](#_Toc41983598)

[6 Технико-экономическое обоснование проекта 65](#_Toc41983599)

[6.1 Общая характеристика разрабатываемого веб-приложения 65](#_Toc41983600)

[6.2 Исходные данные 65](#_Toc41983601)

[6.3 Методика обоснования цены 66](#_Toc41983602)

[6.3.1 Объем веб-приложения 67](#_Toc41983603)

[6.3.2 Основная заработная плата 68](#_Toc41983604)

[6.3.3 Дополнительная заработная плата 68](#_Toc41983605)

[6.3.4 Отчисления в Фонд социальной защиты населения 68](#_Toc41983606)

[6.3.5 Расходы на материалы 69](#_Toc41983607)

[6.3.6 Расходы на оплату машинного времени 69](#_Toc41983608)

[6.3.7 Прочие прямые затраты 70](#_Toc41983609)

[6.3.8 Накладные расходы 70](#_Toc41983610)

[6.3.9 Сумма расходов на разработку веб-приложения 70](#_Toc41983611)

[6.3.10 Расходы на сопровождение и адаптацию 71](#_Toc41983612)

[6.3.11 Полная себестоимость 71](#_Toc41983613)

[6.3.12 Определение цены, оценка эффективности 71](#_Toc41983614)

[6.4 Вывод по разделу 72](#_Toc41983615)

[Заключение 74](#_Toc41983616)

[Список использованной литературы 75](#_Toc41983617)

[ПРИЛОЖЕНИЕ А Блок-схема алгоритма изменения проекта 76](#_Toc41983618)

[ПРИЛОЖЕНИЕ Б Блок-схема алгоритма генерации мобильного приложения 77](#_Toc41983619)

[ПРИЛОЖЕНИЕ В Блок-схема алгоритма добавления элемента на страницу 78](#_Toc41983620)

[ПРИЛОЖЕНИЕ Г Блок-схема функции добавления нового проекта в БД 79](#_Toc41983621)

[ПРИЛОЖЕНИЕ Д Диаграмма последовательности работы приложения 80](#_Toc41983622)

[ПРИЛОЖЕНИЕ Е Исходный код генератора мобильного приложения 81](#_Toc41983623)

# Введение

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

Рыжанкова А.С.

Вольский Д.М.

Жиляк Н.А.

Ф.И.О.

Подпись

Дата

1

*Введение*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*74417012, 2020*

На сегодняшний день тяжело представить жизнь современного человека без «всемирной паутины», предоставляющего доступ к огромному количеству данных. Одним из пользовательских элементов данной системы являются веб-ресурс или же сайт, представляющие совокупность логически связанных между собой веб-страниц. В настоящее время скорость роста в нашем мире количества Веб-ресурсов увеличивается. Для большинства людей намного удобней решать возникающие потребности, не отходя от компьютера.

В современном обществе при текущем быстром ритме жизни, одним из самых драгоценных ресурсов стало время. Веб-приложения помогают людям сохранить свое время, так как для их запуска нужен только браузер и наличие доступа в интернет. Поэтому человек может воспользоваться ресурсом в любое время и в любом месте, в не зависимости от устройства, которое у него есть.

В настоящее время все вокруг стремятся к автоматизации рутинных процессов. Одним из таких процессов для дизайнеров и разработчиков является создание понятных и информативных интерфейсов. Так как понятный разработчику и конечному пользователю интерфейс может сэкономить время разработки приложения, то исходя из этого было разработано веб-приложение которое, может ускорить процесс создания качественных интерактивных интерфейсов.

Целью данного дипломного проекта является создание веб-приложения, которое позволит осуществлять генерацию интерактивных интерфейсов для операционной системы *Android*.

Для достижения поставленной цели необходимо решить следующие задачи:

* разработать структуру интернет-ресурса;
* выборать и обосновать средства разработки;
* разработать пользовательский интерфейс;
* спроектировать базы данных для работы пользователя с интернет-ресурсом для того, чтобы осуществлять контроль над проектами и процессом их создания, редактирования и удаления;
* сгенерировать мобильное приложение на основе созданного проекта.

Консульт.

Жиляк Н.А.

# 1 Обзор теоретического материала и аналогов

ФИО

Подпись

Дата

Лист

1

БГТУ 01.00.ПЗ

Разраб.

*Вольский Д.М.*

Пров.

Жиляк Н.А.

Консульт.

Жиляк Н.А.

Н. контр.

Рыжанкова А.С.

Утв.

Смелов В.В.

1 Обзор теоретического материала и аналогов

Лит.

Листов

8

*74218005, 2020*

## Основные сведения по теме «Прототипирования»

Прототипирование – это деятельность по созданию прототипов программных приложений, то есть неполных версий разрабатываемого программного обеспечения. Это деятельность, которая может происходить при разработке программного обеспечения и сравнима с прототипированием, известным из других областей, таких как машиностроение или производство.

Прототип обычно имитирует лишь несколько аспектов конечного продукта и может полностью отличаться от него.

Прототипирование имеет несколько преимуществ: дизайнер и разработчик программного обеспечения могут получить ценную обратную связь от пользователей в начале проекта. Клиент и подрядчик могут сравнивать, соответствует ли выполненное программное обеспечение спецификации программного обеспечения, в соответствии с которой создается программное обеспечение. Это также позволяет разработчику программного обеспечения получить представление о точности первоначальных оценок проекта и о том, могут ли предлагаемые сроки и этапы быть успешно выполнены.

Цель прототипа состоит в том, чтобы позволить пользователям программного обеспечения оценивать предложения разработчиков по дизайну конечного продукта, фактически испытывая их, вместо того, чтобы интерпретировать и оценивать проект на основе описаний. Прототипирование программного обеспечения обеспечивает понимание функций программного обеспечения и потенциальных угроз или проблем. Прототипирование может также использоваться конечными пользователями для описания и обоснования требований, которые не были рассмотрены и которые могут стать ключевым фактором в коммерческих отношениях между разработчиками и их клиентами. В частности, дизайн взаимодействия интенсивно использует прототипирование с этой целью.

Создание и оценка прототипа позволяют в таких случаях на ранних стадиях разработки отделить важные для заказчика детали, которые не попали в спецификацию, от неважных, реализация которых может быть произвольной.

В разработке систем, основная работа которых состоит в вычислениях, например, программ и утилит с интерфейсом командной строки, прототипирование почти не дает реальных преимуществ.

## Типы и классификации прототипов

В настоящее время в нашей с вами жизни уже создано большое количество различных вариантов прототипирования интерфейсов. Однако, все методы в какой-либо степени построены на двух основных типах. Итак, классифицируем по типам создания прототипов:

* быстрое прототипирование:
* эволюционное прототипирование;.

При [быстром прототипировании](https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%81%D1%82%D1%80%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) предполагается, что создается макет, который на каком-то этапе будет оставлен («выброшен») и не станет частью готовой системы.

Основное преимущество такого подхода – в скорости: в ответ на свои требования заказчик почти сразу получает прототип интерфейса, и сразу может уточнить требования, до того как начато написание рабочего кода системы. Стоимость изменения требований на этом этапе очень низкая, поскольку нет кода, который нужно было бы переписывать.

Очень важно, чтобы такое прототипирование было выполнено в кратчайшие сроки, поскольку в данном случае тратятся время и ресурсы на код, который не будет в дальнейшем использован.

Быстрое прототипирование не обязательно выполняется в рамках той же платформы и тех же технологий, что и разрабатываемая система. Для прототипа [графического интерфейса пользователя](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) (*GUI*) могут использоваться как стандартные [HTML](https://ru.wikipedia.org/wiki/HTML)-страницы, либо прототип может подготавливаться в программе, специально предназначенной для создания макетов (например: [*Axure RP*](https://ru.wikipedia.org/wiki/Axure_RP), [*Microsoft Expression Blend*](https://ru.wikipedia.org/wiki/Microsoft_Expression_Blend) и др.).

Эволюционное прототипирование ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) e*volutionary prototyping*) ставит своей целью последовательно создавать макеты системы, которые будут все ближе и ближе к реальному продукту.

Такой подход имеет то преимущество, что на каждом шаге мы располагаем работающей системой, пусть и не обладающей всей нужной функциональностью, но улучшающейся с каждой итерацией. При этом, не тратятся ресурсы на код, который будет «выброшен».

Эволюционный подход к прототипированию может быть выбран, исходя из предположения, что все необходимые требования к моменту начала разработки неизвестны, и будут определяться по мере создания программы; тогда на каждом этапе мы реализуем лишь те требования, которые известны и ясны. Иногда при этом разработчики сосредотачиваются на работе только над теми модулями системы, требования на которые уже определены.

В некоторых случаях, когда речь идет о продукте под определенную незанятую нишу, пользователи начинают использовать систему еще до того, как она полностью дописана, в ожидании готовой системы, поскольку «недописанная система – это лучше, чем ее полное отсутствие».

## Обзор аналогов

В последние 3-4 лет в мире стремительно набирает темп внедрения онлайн сервисов по созданию интерактивных интерфейсов.

Их суть достаточна проста: вначале пользователь создает пустой проект своего приложения интерфейса, затем, на этот проект он может добавлять нужные элементы, редактировать их, менять местоположение. Затем, когда итерфейс создан, создается веб-страница, на которой отображается визуализация созданного интерфейса мобильного приложения.

Однако у такого подхода есть большая проблема, у всех современных мобильных устройств разрешение и размеры экрана разные. Поэтому подобрать нужный размер для прототипа представляет трудность. Так же отличается отображение элементов в существующих конструкторах прототипов от стандартных элементов в операционной системе *Android*.

Существующие сервисы по созданию прототипов интерфейсов позволяют съэкономить время работы дизайнера.

Раньше у людей не было возможности использовать сервисы по созданию интерактивных интерфейсов, но с развитием технологий и вычислительной мощи появилась возможность в создании таких сервисов, которые ускорят работу дизайнеров и разработчиков в несколько раз. Так как быстро и качественно созданный прототип поможет разработчику сделать свою работу без ошибок и за более короткий срок.

### **1.3.1 Сервис создания динамических интерфейсов «Marvelapp»**

*Marvelapp* – сервис для прототипирования приложений. Сервис помогает сделать постраничный прототип мобильного приложения. Данный сервис имеет возможность закрепить навбар и таббар и листать контент внутри, а плагин *Marvel* для *Sketch* позволяет импортировать макет из *Sketch* в *Marvel*. Но при этом в программе можно создать дизайн при помощи внутренней «дизайн-студии». К преимуществам такого сервиса можно отнести:

* сложные инструменты дизайна;
* доступ к прототипу из любого места.

К основным функциям можно отнести послойное прототипирование, хоть и полноценным его назвать трудно. Сервис отличает красивый интерфейс и яркие цвета. Однако он является платным и распространяется по подписочной модели. Так же имеется и бесплатная подписка, которая позволяет создать один бесплатный проект.

Если же существует необходимость поделиться созданным проектом с другими пользователями приложения, то необходимо просто указать ссылку на созданный интерактивный интерфейс.

Сущетсвенными недостатками данного приложения являются следующие моменты:

* отсутствуют разные состояния экрана мобильного телефона;
* поделиться готовым проектом можно только с пользователем, который зарегистрирован в приложении, что не всегда является удобным. Если бы возможность обмена готовым проектом существовала без ограничений данного сервиса, то этот момент являлся бы значительным преимуществом приложения, ведь таким образом повышается уровень рекламы сервиса, тем самым привлекая все больше пользователей, которые готовы приступить к созданию новых проектов.

Страницу редактирования прототипа рассмотрим на рисунке 1.1.

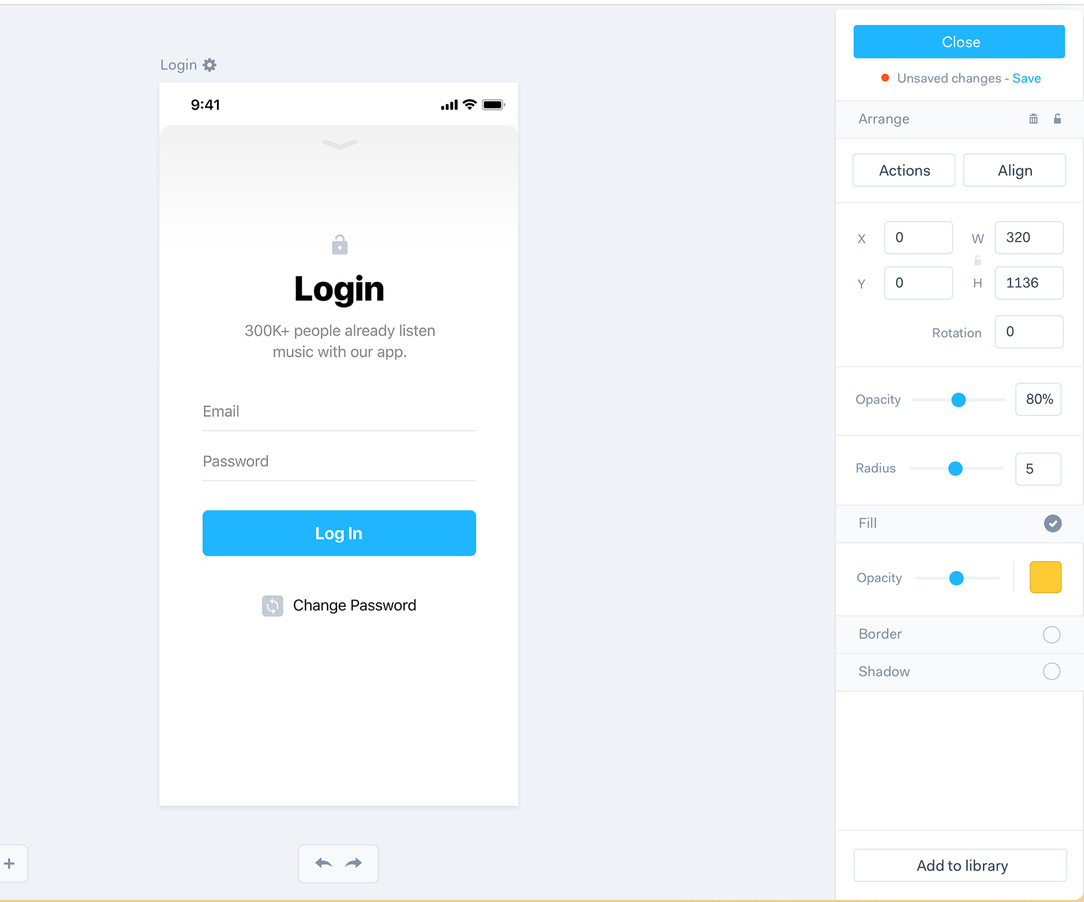


Рисунок 1.1 – Фрагмент сервиса для создания прототипов интерфейсов

Локаничность данного веб-приложения делает его приятным для использования.

### **1.3.2 Сервис создания динамических интерфейсов «InVision»**

*InVision* – программа для прототипирования интерфейса более уссовершенствованная, чем другие сервисы, для создания прототипов интерфейса, и позиционирует она себя как инструмент для профессионалов. У программы есть плагин для *Sketch* – *Craft*, который позволяет превращать дизайн-макет в прототип прямо в *Sketch*, без отрыва от производства. *Sketch* – векторный графический редактор для *macOS*. Оличительными чертами *Invision* являются:

* быстрый процесс подстановки шаблонов;
* быстрая замена имен;
* настройка заголовков;
* замена дат и текстов;
* отзывчивый интерфейс;
* стабильность работы.

Так же, в макете можно быстро представить кучу разных имен, с помощью выделения нужных слоев и нажатия соответствующей кнопки в плагине, в это время плагин сам подставляет нужный контент. Этот же плагин дает возможность использовать фото с бесплатного стока фотографий *Unsplash*. *Unsplash* – это веб-сайт, посвященный обмену фотографиями по лицензии. Что позволяеет пользователю сократить огромное количество времени для разработки интерактивного интерфецса своего мобильного, а так же десктопного приложения, которое очень важно для разработчиков, а так же дизайнеров.

Пример создания прототипа интерфейса изображен на рисунке 1.2.

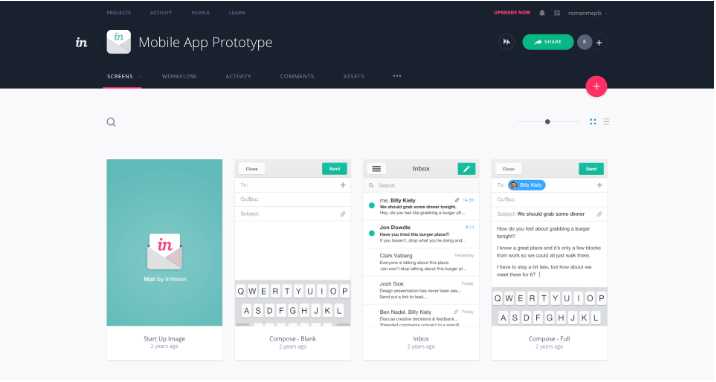


Рисунок 1.2 – Экран проекта в «InVision»

При создании нового прототипа, сервис предлагает выбрать роль для того, кто будет редактировать макет, и исходя из этой роли в панель инструментов будут добавлены специфические инструменты редактирования элементов. Так же при созданиии прототипа можно выбрать для какой платформы будет разрабатываться интерфейс: мобильное приложение, десктопное приложение или приложение для носимых устройств.

Существенным недостатком сервиса является отсутствие добавления стандартных элементов операционой системы. Так же недостатком является интуитивно не понятный интерфейс.

К плюсам сервиса можно отнести следующие:

* удобные комментарии к прототипам
* разработчик может посмотреть размеры объектов в прототипе
* есть возможность совместной работы и менеджер задач
* сервис абсолютно бесплатный

### **1.3.3 Сервис создания динамических интерфейсов «Origami Studio»**

«Origami Studio» – программа для разработки интерактивных интерфейсов приложений. Приложение работает только на операционной системе *MacOs*. К программе можно подключить очень много патчей (приложений, которые расширяют функционал существующей программы), что является ее отличительной чертой, но разобраться в них рядовому пользователю достаточно сложно.

Эта программа, в отличие от предыдущих, послойная. В ней можно создавать анимации и эффектные переходы между экранами, показывать клиенту или пользователю, как элементы реагируют на обращение к ним. Этот тип программ больше подходит для небольших приложений, максимум на пять экранов. Продумывать все взаимодействия для 50 экранов очень долго, и вряд ли это стоит того на прототипе. Пример создания прототипа можно увидеть на рисунке 1.3

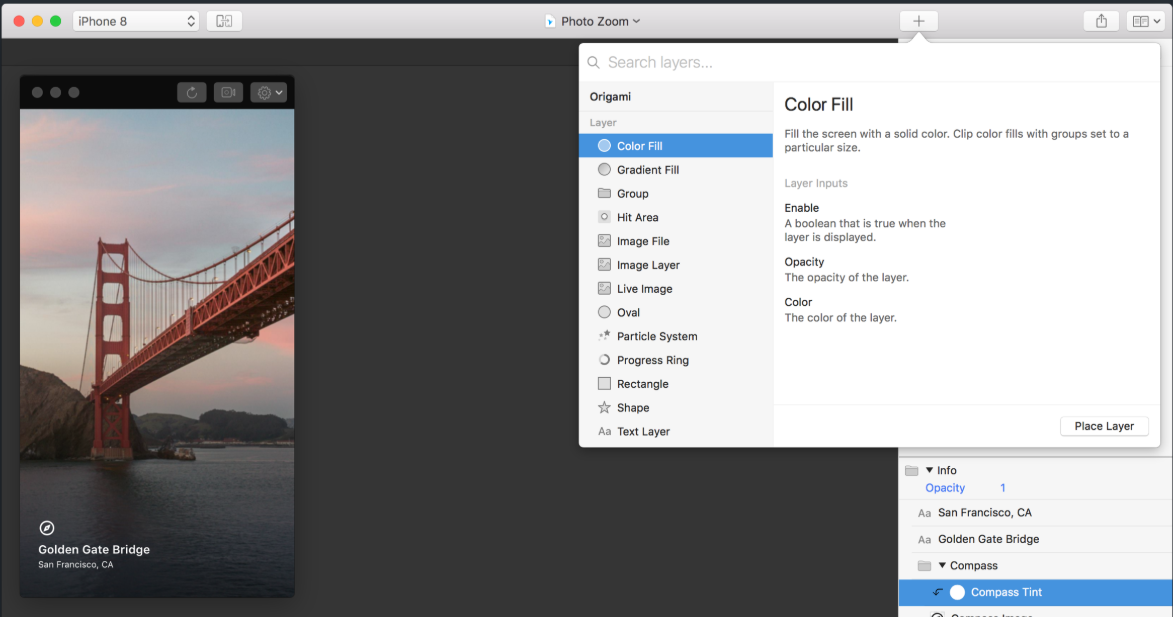


Рисунок 1.3 – Экран проекта в Origami Studio

Окно создания прототипа состоит из четырех частей: панели слоев, предпросмотра, окна редактирования, библиотеки элементов.

Программа имеет прямую привязку к операционной системе. Из-за этого, просмотреть прототип могут только те пользователи, у которых есть компьютер на платформе *MacOs*, а так же есть аккаут разработчика.

К плюсам программы можно отнести то, что она бесплатна. Так же плюсом будет то, что интерфейс здесь интуитивно понятный, а так же, есть большое количество документации для каждого интрумента.

## Патентный поиск по теме дипломного проекта

В соответствии с темой дипломного проекта был проведен патентный поиск в области программных систем для создания интерфейса.

Патентный поиск – это процесс отбора соответствующих запросу документов или сведений по одному или нескольким признакам из массива патентных документов или данных, при этом осуществляется процесс поиска из множества документов и текстов только тех, которые соответствуют теме или предмету запроса.

Патентный поиск осуществляется посредством информационно-поисковой системы и выполняется вручную или с использованием соответствующих компьютерных программ, а также с привлечением соответствующих экспертов.

Предмет поиска определяют исходя из конкретных задач патентных исследований категории объекта (устройство, способ, вещество), а также из того, какие его элементы, параметры, свойства и другие характеристики предполагается исследовать.

При патентном поиске сравниваются выражения смыслового содержания информационного запроса и содержания документа.

Изобретение – это основной объект промышленной собственности.

Патент на изобретение – это документ, выдаваемый компетентным государственным органом и удостоверяющий: приоритет изобретения, авторство и исключительное право на изобретение. Действует в пределах территории того государства, ведомство которого его выдало.

Под патентоспособностью понимают юридическое свойство объекта промышленной собственности, определяющее его способность охраняться документом исключительного права (патентом) на территории конкретной страны в течение срока действия патента.

Решение является патентоспособным если:

* это способ или вещество;
* обладает новизной;
* имеет изобретательский уровень;
* является промышленно применяемым.

Патентный поиск – это процесс отбора соответствующих запросу документов или сведений по одному или нескольким признакам из массива патентных документов или данных, при этом осуществляется процесс поиска из множества документов и текстов только тех, которые соответствуют теме или предмету запроса.

В качестве аналогов можно выделить: Веб-приложение-генератор для мобильных устройств с автоматическим обновлением.

### **1.4.1 Веб-приложение-генератор для мобильных устройств**

Язык программирования: С#.

Номер регистрации (свидетельства): EP2702485A1.

Дата регистрации: 26.04.2011.

Реферат:

Изобретение позволяет пользователям создавать, настраивать, управлять и обновлять свои собственные мобильные приложения через механизм создания онлайн-приложений. Пользователь собирает приложение из стандартизированных модулей - строительных блоков, поставляемых в механизме создания приложений, - выбирая модули с желаемой функциональностью. Процесс создания приложения не требует кодирования. Кроме того, поскольку модули могут находиться на сервере, приложение может быть изменено или обновлено автоматически путем изменения приложения или отдельных модулей на сервере. Затем обновления автоматически включаются в приложение после запуска приложения на устройстве пользователя, таком как смартфон или планшет.

## Выводы по разделу

В основном, оценка ресурса по генерации интерактивных интерфейсов для операционной системы *Android*, сводится к удобству пользовательского интерфейса, хорошей автоматизации и скорости работы, а так же кроссплатформенность. В данном разделе был произведен обзор аналогов, рассмотрены их плюсы и минусы, произведен патентный поиск аналогов программного обеспечения по теме дипломного проекта.

Каждый аналог был подробно описан, и было показано, для чего предназначено то или иное программное обеспечение.

Как видно из представленного краткого обзора, большинство ресурсов имеют много общего в своих функциональных возможностях. В первую очередь это касается базовых функций: создание проекта прототипа, возможность редактирования существующего прототипа, а так же возможность поделиться готовым интерактивным прототипом.

При более детальном рассмотрении аналогов приложения становится понятно, что одни из ресурсов не имеют в своем арсенале удобного и понятного для пользователя интерфейса, а другие не могут дать возможностей просмотра готового интерактивного интерфейса на той платформе, для которой создавался этот прототип или не имеют удобного способа передачи готового прототипа заказчику.

# 2 Технические средства разработки

ФИО

Подпись

Дата

Лист

1

БГТУ 02.00.ПЗ

Разраб.

*Вольский Д.М.*

Пров.

Жиляк Н.А.

Консульт.

Жиляк Н.А.

Н. контр.

Рыжанкова А.С.

Утв.

Смелов В.В.

2 Технические средства разработки

Лит.

Листов

11

*74218005, 2020*

Перед началом разработки необходимо определить цели, задачи, все варианты использования программного средства. Для этого необходимо построить диаграмму использования.

Диаграмма вариантов использования приложения, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы (часть ее функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

Основное назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

В результате была разработана диаграмма вариантов использования. С ее использованием будет проходить дальнейшая разработка функциональных возможностей веб-приложения. Диаграмма вариантов использования представлена на рисунке 2.1.



Рисунок 2.1 – Диаграмма вариантов использования приложения

Из диаграммы вариантов использования видно действующих лиц, их взаимодействие с системой и ожидаемую функциональность системы. С ее использованием будет проходить дальнейшая разработка веб-приложения.

Также перед началом разработки необходимо установить нужные программные средства для разработки. Поэтому первым в процессе разработки является этап настройки среды разработки.

Далее следует этап проектирования визуальных компонент, которые будут непосредственно отображать информацию, получаемую в результате работы веб-приложения. Под визуальными компонентами в первую очередь понимается дизайн, а также поля, где будет осуществляться вывод и ввод информации.

На следующем этапе нужно разработать систему обеспечивающую функционал работы веб-приложения.

На рисунке 2.2 и в приложении Д представлена диаграмма последовательности, показывающая последовательность действий, выполняемых системой.

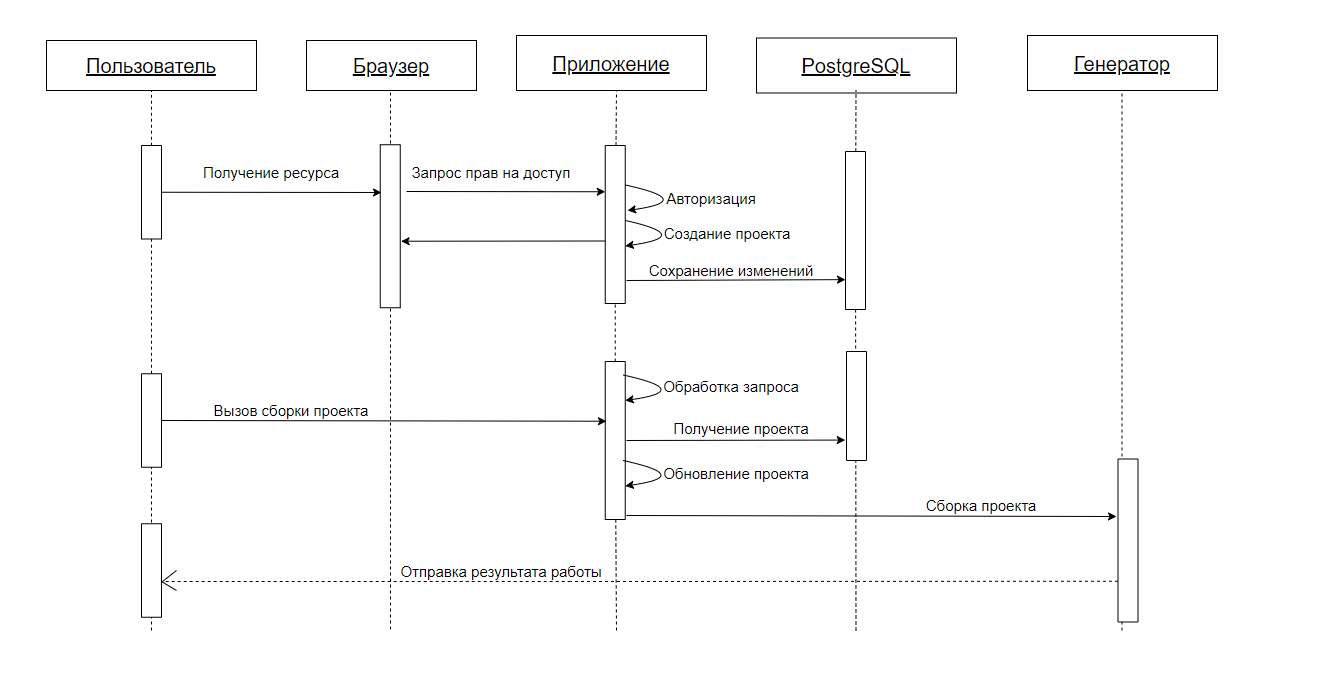


Рисунок 2.2 – Диаграмма последовательности работы приложения

Диаграмма последовательностей очень похожа на диаграмму взаимодействий по тому, какую информацию они показывают. Главное отличие диаграммы последовательностей в том, что на этой диаграмме легче проследить последовательность действий. Кроме того, на этой диаграмме можно указать подробную информацию о времени существования того или иного объекта или о его поведении (например, время ожидания, взаимодействие параллельных нитей процесса, момент создания и уничтожения объектов).

## Описание работы и структуры программного средства

Основной частью приложения должна являться страница – констуктор. При первом посещении пользователь попадет на страницу авторизации. После успешной авторизации пользователь будет переходить на страницу списка проектов. Здесь он сможет выбрать существующий проект, изменить, удалить его, создать новый. При нажатии на кнопку «создать новый проект» откроется страница редактора. Пользователь сможет заполнить основные настройки проекта: название, иконка приложения, основные цвета. Затем пользователь может приступить к созданию самого приложения.

Приложение будет состоять из страниц (экранов). Существует два типа страниц: обычные и всплывающие. Сперва пользователь создаст страницу, даст ей название, выберет ее тип, а затем может добавлять на нее контроллы (элементы). Список контролов будет расположен в блоке, который открывается по клику на название в меню. Пользователь сможет перетащить контрол из блока на сетку, при отпускании контрол останется в ее пределах. Контрол будет отображаться только в пределах сетки.

В блоке будет присутствовать поиск, при вводе в строку поиска будет происходить фильтрация и отображение только тех контролов, в названии которых присутствуют набранные сочетания символов. Также при начале ввода в строку в правом углу появится знак удаления (крестик), при нажатии на который введенный текст удаляется.

При выделении контрола на сетке по клику справа будет отображаться панель настроек контрола. Также на контроле будут отображаться элементы для изменения размера и поворота. Для осуществления поворота необходимо будет либо потянуть за элемент поворота, либо ввести значение в поле на панели редактирования.

Удаление контрола будет происходить либо при нажатии на иконку удаления на панели редактирования, либо при нажатии на клавишу «Delete».

Дублирование элемента будет происходить двумя способами: при нажатии на иконку копирования на панели редактирования, либо при нажатии на сочетание клавиш Ctrl+V Ctrl+C.

Основой взаимодействия между страницами является элемент «кнопка», так как этому элементу можно задать свойство действия (переход по ссылке, открытие страницы, вернуться назад). После добавления элементов на страницу можно открыть превью – режим. В небольшой форме отобразится внешний вид создаваемого приложения с кликабельными кнопками. После настройки приложения пользователем, приложение можно сгенерировать. При нажатии на кнопку «генерация» будет отправляться запрос на генератор и в течение двух минут будет собираться *apk* версия, далее происходит ее скачивание.

## Обоснование выбора языка программирования

C# – язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Разработан в 1998–2001 годах группой инженеров под руководством Андерсa Хейлсбергa в компании *Microsoft* как основной язык разработки приложений для платформы *Microsoft .NET*. Компилятор с C# входит в стандартную установку самой .*NET*, поэтому программы на нем можно создавать и компилировать даже без инструментальных средств вроде *Visual Studio.*

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и *Java*. Язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате *XML*. Переняв многое от своих предшественников – языков C++, *Delphi, Modula* и *Smalltalk* – С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++) или вывода типов (в отличие от *Haskell*). С помощью языка C# можно создавать обычные приложения *Windows*, *XML*-веб-службы, распределенные компоненты, приложения «клиент-сервер», приложения баз данных и т. д.

Исходя из перечисленных преимуществ, для разработки веб-приложения был выбран язык C#.

## Обзор средств разработки

### **2.3.1 Платформа .NET**

*.NET Framework* – это программная платформа, выпущенная компанией Microsoft, которая подходит для разных языков программирования [1].

.*NET* достаточно широко распространен в сфере разработки внутрикорпоративных программных продуктов, но в веб-разработке все же встречается относительно редко, как и другие программные продукты корпорации *Microsoft*. Поэтому найти разработчиков для веб-проекта бывает достаточно непросто. Использование. *NET* «тянет» за собой покупку и иного ПО от корпорации *Microsoft* (серверной ОС, СУБД и т.п.). Технология достаточно дорогая в разработке и сопровождении: кроме затрат на покупку лицензий на необходимое ПО существенный вклад в бюджет проектов вносят высокие зарплаты разработчиков.

*ASP.NET Core* – представляет технологию от компании *Microsoft*, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов..

### **2.3.2 Платформа ASP.NET Core**

*ASP.NET Core* – севободно-распространяемый кросс-платформенный фреймворк для создания веб-приложений с открытым исходным кодом. Данная платформа разрабатывается компанией Майкрософт совместно с сообществом и имеет большую производительность по сравнению с *ASP.NET*. Имеет модульную структуру и совместима с такими операционными системами как *Windows*, *Linux* и *MacOS*.

*ASP.NET Core* включает в себя фреймворк *MVC*, который объединяет функциональность *MVC* [2], *Web API* и *Web Pages*. В предыдущих версии платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель *ASP.NET Core MVC*. А *Web Forms* полностью ушли в прошлое. Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер *Nuget*. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку *System.Web.dll*. Некоторые особенности *ASP.NET* [3]:

* распространение пакетов платформы через *NuGet*;
* интегрированная поддержка для создания и использования пакетов *NuGet*;
* единый стек веб-разработки, сочетающий *Web UI* и *Web API*;
* конфигурация для упрощенного использования в облаке;
* встроенная поддержка для внедрения зависимостей;
* поддержка *CRUD*-операций при работе с таблицами через *GridView*;
* возможно создание веб-приложений, которые реализуют шаблон *Model-View-Controller* (*ASP.NET MVC Framework*).

### **2.3.3 Платформа ASP.NET Core MVC**

Одним из отличительных моментов платформы *ASP.NET Core* является применение паттерна *MVC*. Причем последняя версия *MVC*-фреймворка, который применяется в *ASP.NET Cor*e. Поэтому важно не путать *ASP.NET MVC 5*, который применяется в *ASP.NET*, и фреймворк *MVC*, который применяется в *ASP.NET Core.* Хотя во многих аспектах эти фреймворки будут совпадать.

Если оперировать понятиями высокого уровня, архитектурный шаблон *MVC* означает, что приложение *MVC* будет разделено, по крайней мере, на три части (рисунок 2.3).



Рисунок 2.3 – Структурные части архитектурного шаблона MVC

Модели, содержащие или представляющие данные, с которыми работают пользователи. Они могут быть простыми моделями представлений, которые только представляют данные, передаваемые между представлениями и контроллерами; или же они могут быть моделями предметной области, которые содержат бизнес – данные, а также операции, преобразования и правила для манипулирования этими данными.

Представления, применяемые для визуализации некоторой части модели в виде пользовательского интерфейса.

Контроллеры, которые обрабатывают поступающие запросы, выполняют операции с моделью и выбирают представления для визуализации пользователю.

В *MVC* контроллеры являются классами, обычно производными от класса *System.Web.Mvc.Controller* [4]. Каждый метод public в классе, унаследованном от класса *Controller*, называется методом действия и посредством системы маршрутизации *ASP.NET* связан с конфигурируемым *URL*. Когда запрос отправляется *URL*, связанному с методом действия, операторы в классе контроллера выполняются, чтобы провести некоторую операцию по отношению к модели предметной области и затем выбрать представление для отображения клиенту. Взаимодействия между контроллером, моделью и представлением показаны на рисунке 2.4.

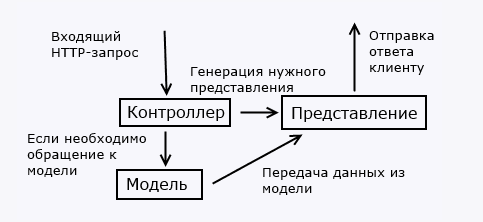


Рисунок 2.4 – Взаимодействия между контроллером, моделью и представлением

Платформа *ASP.NET MVC* предоставляет поддержку для выбора механизмов визуализации [5]. В более ранних версиях *MVC* использовался стандартный механизм визуализации *ASP.NET*, который обрабатывал *ASPX*-страницы с применением оптимизированной версии синтаксиса разметки *Web Forms*. В платформе *MVC 3* был введен механизм визуализации *Razor*, который использует совершенно другой синтаксис [6]. *Visual Studio* обеспечивает поддержку средства *IntelliSense* для обоих механизмов визуализации, максимально упрощая внедрение и ответ на данные представления, предоставленные контроллером.

### **2.3.4 База данных PostgreSQL**

Чтобы хранить данные, нам естественным образом нужна база данных. Как правило, в качестве базы данных используется *PostgreSQL*. Для управления базой данных необходимо установить *pgAdmin*. Так же оно нужно для проверки корректности информации и структуры базы данных. Это может быть информация любого вида:

* информация о пользователях;
* временное значение переменных;
* начальные значения;
* связи объектов.

Также необходимо сказать об файле для отправки генератору. Так как *xml* фаил может быть большого размера, то требуется структуированный формат данных. В качестве шаблона был выбран *JSON* файл. Выбор пал на данный формат, так как для многих языков существует готовый код для создания и обработки данных в формате *JSON*, он не содержит лишних символов, а так же он легко читаем.

### **2.3.5 Структурированный язык запросов SQL**

*SQL* (*Structured Query Language* – Структурированный язык запросов) – язык управления базами данных для реляционных баз данных. Сам по себе *SQL* не является тьюринг-полным языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые расширяют его функциональность до полноценного языка программирования [7].

Язык был создан в 1970х годах под названием «*SEQUEL*» для системы управления базами данных (СУБД) *System R*. Позднее он был переименован в «SQL» во избежание конфликта торговых марок. В 1979 году *SQL* был впервые опубликован в виде коммерческого продукта *Oracle V2*.

*SQL* создавался как простой стандартизированный способ извлечения и управления данными, содержащимися в реляционной базе данных. Позднее он стал сложнее, чем задумывался, и превратился в инструмент разработчика, а не конечного пользователя. В настоящее время SQL (по большей части в реализации Oracle) остается самым популярным из языков управления базами данных, хотя и существует ряд альтернатив.

*SQL* состоит из четырех отдельных частей.

Язык определения данных (*DDL*) используется для определения структур данных, хранящихся в базе данных. Операторы *DDL* позволяют создавать, изменять и удалять отдельные объекты в БД. Допустимые типы объектов зависят от используемой СУБД и обычно включают базы данных, пользователей, таблицы и ряд более мелких вспомогательных объектов, например, роли и индексы.

Язык манипуляции данными (*DML*) используется для извлечения и изменения данных в БД. Операторы *DML* позволяют извлекать, вставлять, изменять и удалять данные в таблицах. Иногда операторы select извлечения данных не рассматриваются как часть *DML*, поскольку они не изменяют состояние данных. Все операторы DML носят декларативный характер.

Язык определения доступа к данным (*DCL*) используется для контроля доступа к данным в БД. Операторы *DCL* применяются к привилегиям и позволяют выдавать и отбирать права на применение определенных операторов *DDL* и *DML* к определенным объектам БД.

Язык управления транзакциями (*TCL*) используется для контроля обработки транзакций в БД. Обычно операторы *TCL* включают commit для подтверждения изменений, сделанных в ходе транзакции, *rollback* для их отмены и savepoint для разбиения транзакции на несколько меньших частей.

### **2.3.6 Технология Entity Framework Сore**

*Entity Framework Core* (EF Core) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании *Microsoft* для доступа к данным. *EF Core* является *ORM*-инструментом (*object-relational mapping* - отображения данных на реальные объекты) [8]. То есть *EF Core* позволяет работать базами данных, но представляет собой более высокий уровень абстракции: *EF Core* позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает *Entity Framework*, мы уже работаем с объектами. Центральной концепцией *Entity Framework* является понятие сущности или *entity*. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами. *Entity Framework Core* поддерживает множество различных систем баз данных. Таким образом, мы можем через *EF Core* работать с любой СУБД, если для нее имеется нужный провайдер.

По умолчанию на данный момент *Microsoft* предоставляет ряд встроенных провайдеров: для работы с *MS SQL Server*, для *SQLite*, для *PostgreSQL*. Также имеются провайдеры от сторонних поставщиков, например, для *MySQL*.

Также стоит отметить, что EF Core предоставляет универсальный API для работы с данными. И если, к примеру, мы решим сменить целевую СУБД, то основные изменения в проекте будут касаться прежде всего конфигурации и настройки подключения к соответствующим провайдерам. А код, который непосредственно работает с данными, получает данные, добавляет их в БД и т.д., останется прежним.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой *Entity Framework Core* является использование запросов *LINQ* для выборки данных из БД. С помощью *LINQ* мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является *Entity Data Model*. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

*Entity Data Model* состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

*Entity Framework* предполагает три возможных способа взаимодействия с базой данных:

– *database first: Entity Framework* создает набор классов, которые отражают модель конкретной базы данных;

– *model first*: сначала разработчик создает модель базы данных, по которой затем *Entity Framework* создает реальную базу данных на сервере;

– *code firs*t: разработчик создает класс модели данных, которые будут храниться в бд, а затем *Entity Framework* по этой модели генерирует базу данных и ее таблицы.

### **2.3.7 Язык программирования JavaScript**

*JavaScript* – это полноценный динамический язык программирования, который применяется к *HTML* документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал *Brendan Eich*, сооснователь проекта *Mozilla, Mozilla Foundation* и *Mozilla Corporation* [9].

*JavaScript* довольно универсален. Работу с данным языком можно начать с малого, с простых функций, таких как карусели, галереи изображений, изменяющиеся макеты и отклик на нажатие кнопок [10]. Обладая большим опытом уже можно создавать игры, анимированную 2D и 3D графику, полномасштабные приложения с базами данных и многое другое.

*JavaScript* сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка *JavaScript*, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

– программные интерфейсы приложения (*API*), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание *HTML* и установку *CSS* стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов;

– сторонние *API* позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как *Twitter* или *Facebook*;

– также есть возможность применить к *HTML* сторонние фреймворки и библиотеки, что позволит ускорить создание сайтов и приложений.

### **2.3.8 Библеотека jQuery**

*jQuery* – это *JavaScript*-библиотека, фокусирующаяся на взаимодействии *JavaScript*, *HTML* и *CSS* [11].

Возможности *jQuery*:

– обращение к любому элементу *dom* (объектной модели документа) и не только обращение, но и манипулирование ими;

– работа с событиями;

– осуществление различных визуальных эффектов;

– работать с ajax (технология, позволяющая общаться с сервером без перезагрузки страницы);

– имеет огромное количество JavaScript плагинов, предназначенных для создания элементов пользовательских интерфейсов.

Особенность того, почему jQuery стала настолько успешной и популярной, это, вероятно, кроссплатформенные возможности. Она автоматически исправляет ошибки и работает таким же образом в наиболее часто используемых браузерах, таких как *Chrome, Firefox, Safari, MS Edge, IE, Android* и *iOS* [12].

*JQuery* также делает *Ajax* намного проще. *Ajax* работает асинхронно с остальной частью кода. Это означает, что код, написанный на *Ajax*, может взаимодействовать с сервером и обновлять его содержимое без необходимости перезагрузки страницы.

Однако это связано с проблемами. Различные браузеры выполняют *Ajax API* по-разному. Таким образом, код должен соответствовать всем браузерам. Вручную, это тяжелая и трудоемкая работа. К счастью, *jQuery* выполняет всю тяжелую работу и адаптирует код для всех веб-браузеров [13].

Затем есть манипулирование *DOM* (*Document Object Model*), в котором есть несколько методов, как это сделать. Проще говоря, он позволяет вставлять и/или удалять элементы *DOM* на *HTML*-странице, а также упрощает перенос строк.

Создание анимации также упрощено с помощью *jQuery*. Как и в приведенном выше фрагменте кода об анимации, он покрыт несколькими строками кода, все что вам нужно сделать, это вставить переменные.

### **2.3.9 Платформа Xamarin.Forms**

*Xamarin.Forms* представляет платформу, которая нацелена на создание кроссплатформенных приложений под *Android*, *iOS* и *Windows* 10 Есть определенные статистические данные, что значительная часть мобильных приложений создается более чем для одной платформы, например, для *Android* и *iOS*. Однако неизбежно разработчики сталкиваются со следующими трудностями:

* различие в подходах построение графического интерфейса так или иначе влияет на разработку. Разработчики вынуждены подстраивать приложение под требования к интерфейсу на конкретной платформе;
* разные *API* - различие в программных интерфейсах и реализациях тех или иных функциональностей также требует от программиста учет этих специфических особенностей;
* разные платформы для разработки. Например, чтобы создавать приложения для *iOS* нам необходима соответствующая среда - *Mac OS X* и ряд специальных инструментов, типа *XCode*.

Такой диапазон платформ, средств разработки и языков программирования не может положительно сказываться на сроках создания приложений, и, в конечном счете, на денежных средствах, выделяемых на разработку. Было бы очень эффективно иметь один инструмент, который позволял легко и просто создавать приложения сразу для всех платформ. И именно таким инструментом и является платформа *Xamarin* (произносится как "зэмарин").

*Xamarin* позволяет создавать одну единcтвенную логику приложения с применением C# и .*NET* сразу для всех трех платформ - *Android*, *iOS*, *UWP*. Преимущества использования *Xamarin.Forms*:

* в процессе разработки создается единый код для всех платформ;
* *xamarin* предоставляет прямой доступ к нативным *API* каждой платформы;
* при создании приложений мы можем использовать платформу .*NET* и язык программирования C# (а также F#), который является достаточно производительным, и в тоже время ясным и простым для освоения и применения;
* *xamarin Forms* поддерживает несколько платформ. Основные платформы: *Android, iOS, UWP, Tizen*.

## Выводы по разделу

В разделе проектирования были рассмотрены основные технологии, необходимые для разработки дипломного проекта. После изучения всех преимуществ этих технологий, для разработки программного средства были выбраны языки программирования С# и *JavaScript* для написания веб-приложения. Данные языки способствуют качеству написания кода, отличаются широким спектром операционных систем, которые могут быть использованы для разработки, позволяют разрабатывать приложение с высоким уровнем быстродействия, а также отвечают всем современным стандартам качества разработки. В целях оптимизации, расширения функциональных возможностей и улучшению качества кода и продукта в целом, были использованы следующие платформы, технологии и библиотеки:

* ASP.NET Core – для создания и веб-приложения с помощью HTML, CSS и JavaScript;
* ASP.NET Core MVC – для разработки веб-приложения, применяя эффективность и аккуратность архитектуры «модель-представление-контроллер»;
* Entity Framework Core – для работы с базами данных через объектно-ориентированный код C#;
* Xamarin.Forms – для генерации мобильного приложения.

В качестве СУБД был выбран *PostgreSQL*, позволяющий в рамках проекта взаимодействовать с приложением. В рамках работы могут использоваться любые версии сервера, так как в *Entity Framework Core*, встроена работа со всеми вариантами данной базы данных.

# Разработка программного обеспечения веб-приложения

ФИО

Подпись

Дата

Лист

1

БГТУ 03.00.ПЗ

Разраб.

*Вольский Д.М.*

Пров.

Жиляк Н.А.

Консульт.

Жиляк Н.А.

Н. контр.

Рыжанкова А.С.

Утв.

Смелов В.В.

3 Разработка программного обеспечения веб-приложения

Лит.

Листов

26

*74218005, 2020*

* 1. Проектирование программного средства

Главной задачей данного дипломного проекта является создание веб-приложение для генерации интерактивных прототипов приложений для операционной системы *Android*. Исходя из изучения существующих аналогов были поставлены задачи по созданию программных компонентов для работы дизайнеров и разработчиков в качестве пользователей, а также по созданию интуитивно понятного интерфейса. Также была поставлена задача по программной реализации ключевого преимущества проекта по сравнению с аналогами - возможности скачивания прототипа в виде мобильного приложения, для дальнейшего просмотра и оценки интерфейса.

Для разрабатываемой системы была выбрана одна из самых распространенных архитектур – трехуровневая архитектура, которая является разновидностью многоуровневой N-*Layer* архитектуры. Строение архитектуры представлено на рисунке 3.1. Многоуровневая архитектура – это клиент-серверная архитектура, в которой функции представления, обработки и хранения данных максимально независимы друг от друга, что облегчает масштабирование проекта.

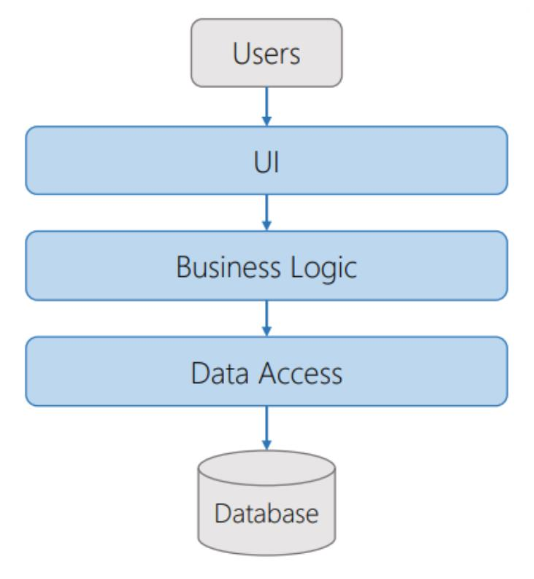


Рисунок 3.1 – Схема трехуровневой архитектуры

Функционально веб-ресурс должен выполнять следующие задачи:

* работа с пользователями (авторизация и регистрация);
* создание, удаление и редактирование проекта;
* создание и настройка динамического прототипа;
* генерация мобильного приложения с созданным интерфейсом.

Разрабатываемое приложение представляет собой клиент-серверное приложение.

Клиент-серверное приложение – это приложение, в котором клиент осуществляет взаимодействие с сервером при помощи браузера. Логика такого распределяется между клиентом и сервером.

## Файловая структура

Перед началом разработки была продумана файловая структура проекта в соответствии с паттерном проектирования *MVC*, позволившая быстрее ориентироваться в коде, вносить изменения, дописывать новый функционал. Схема файловой структуры представлена на рисунке 3.2

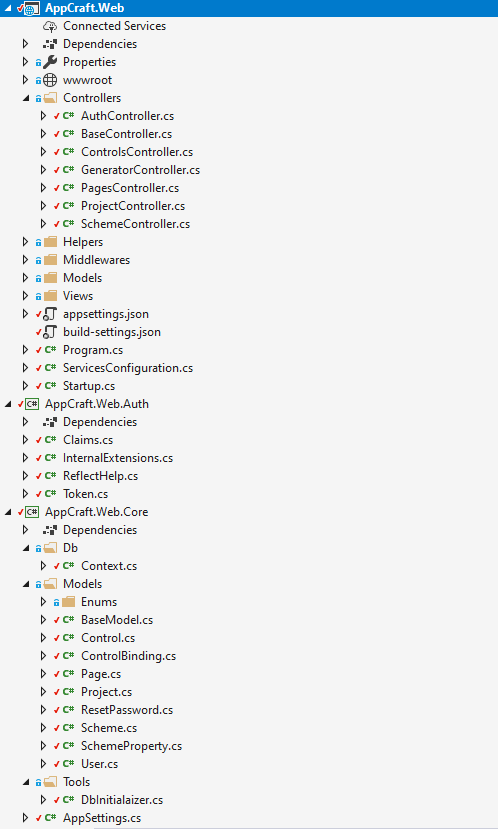


Рисунок 3.2 – Файловая структура веб-приложения

Основное решение проекта содержит большое количество папок, каждая из которых имеет свое назначение. В папке *Models* описаны все классы-модели, которые заполняются при вводе пользователем информации со страницы. В папке *Controllers* описаны методы взаимодействия пользователя с базой данных или для получения нужной ему информации. Папка *wwwroot* используется для хранения всех стилей и изображений, используемых веб-ресурсом. В папке Views хранятся макеты страниц для каждого из контроллера, которые будут получены браузером пользователя при поступлении запроса на контроллер. «Program.cs» – фаил, в котором определен одноименный класс *Program* и с которого по сути начинается выполнение приложения, в данном файле производится инициализация веб-сервера для развертывания веб-приложения. В файле «Startup.cs» содержится одноименный класс *Startup* является входной точкой в приложение *ASP.NET Core*. Этот класс производит конфигурацию приложения, настраивает сервисы, которые прилож­­­­ение будет использовать, устанавливает компоненты для обработки запроса или *middleware*.

## Структура базы данных

В результате проектирования проекта была создана база данных, которая удовлетворяет поставленным задачам. Данная реализация достаточно проста и универсальна, с ее помощью в дальнейшем можно изменять и дополнять структуру базы данных. Структура таблиц базы данных совпадатает со структурой классов-моделей созданных в проекте. Диаграмма их связей представлена на рисунке 3.3

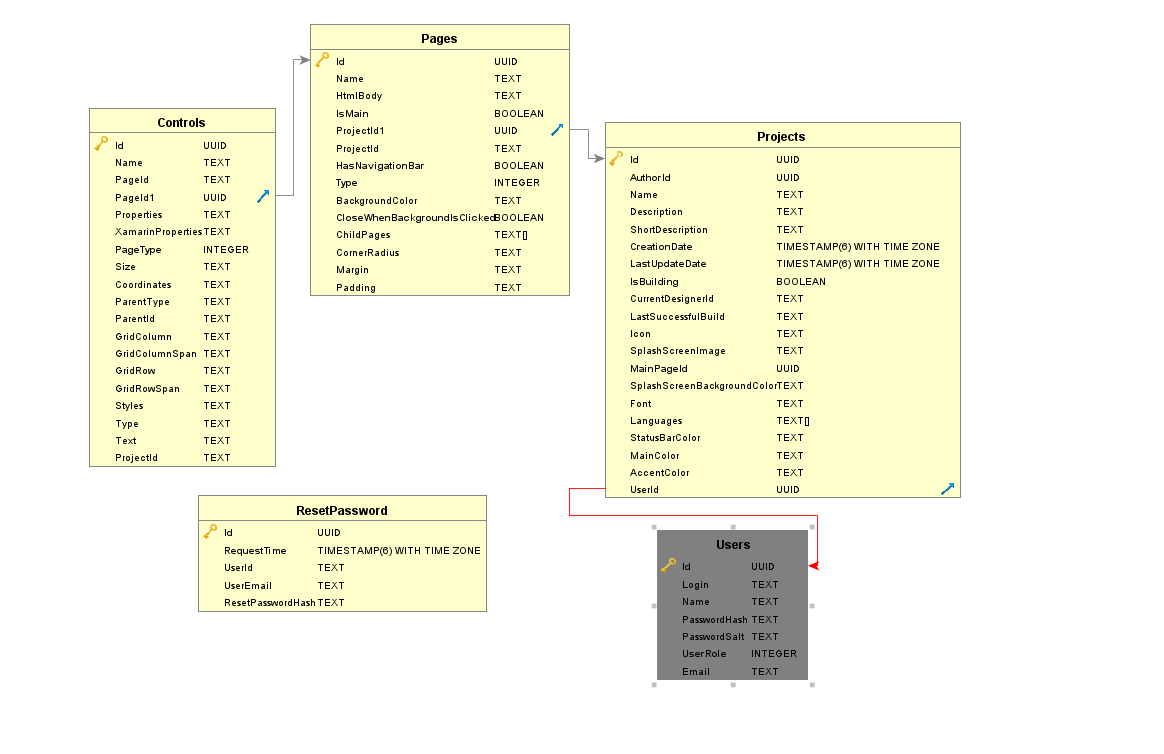


Рисунок 3.3 – Диаграмма связей таблиц

Для реализации функций, которые были определены на стадии проектирования, были созданы следующие таблицы:

* *pages* (информация о созданных страницах);
* *controls* (информация о созданных элементах управления);
* *users* (данные о пользователях);
* *projects* (информация о созданных проектах);
* *resetPassword* (информация для восстановления пароля пользователя).

Для реализации функционала веб-приложения было создано шесть таблиц. Далее приведена их структура и описание.

Таблица «Pages» хранит всю информацию о создаваеммых пользователем страницах и содержит: идентификатор, название страницы, верстка страницы, является ли страница главной, индефикатор проекта, есть ли бар навигации, тип страницы, цвет фона, закрытие страницы по нажатию, индефикаторы дочерних страниц, радиус закругления, внешний отступ, внутренний отступ . Описание ее полей приведено в таблице 3.1.

Таблица 3.1 – Структура таблицы «Pages»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *text* | Идентификатор страницы |
| *name* | *text* | Название страницы |
| *htmlBody* | *text* | Верстка страницы |
| *isMain* | *boolean* | Является ли страница главной |
| *projectId* | *text* | Индефикатор проекта |
| *hasNavigationBar* | *boolean* | Есть ли бар навигации |
| *type* | *integer* | Тип страницы |
| *backgroundColor* | *text* | Цвет фона |
| *closeWhenBackgroundClicked* | *boolean* | Закрытие страницы по нажатию |
| *childPages* | *Text* | Индефикаторы дочерних страниц |
| *cornerRadius* | *text* | Радиус закругления |
| *margin* | *text* | Внешний отступ |
| *padding* | *text* | Внутренний отступ |

Таблица «Projects» отвечает за хранение информации о созданных проектах, она содержит: идентификатор проекта, название проекта, индефикатор пользователя, описание, краткое описание, дата создания, дата последнего изменения, происходит ли сборка проекта, идентификатор пользователя, который вызвал сборку проекта, ссылка на последнюю сборку, фон проекта, шрифт, языки, цвет статус-бара, главный цвет, дополнительный цвет.

Индефикатор пользователя. Описание ее полей приведено в таблице 3.2.

Таблица 3.2 – Структура таблицы «Files»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *text* | Идентификатор проекта |
| *name* | *text* | Название проекта |
| *authorId* | *text* | Индефикатор пользователя |
| *description* | *text* | Описание |
| *shortDescription* | *text* | Краткое описание |
| *creationDate* | *date* | Дата создания |
| *lastUpdateDate* | *date* | Дата последнего изменения |
| *isBuilding* | *boolean* | Происходит ли сборка проекта |
| *currentDesignerId* | *text* | Идентификатор пользователя, который вызвал сборку проекта |

Окончание таблицы 3.2

|  |  |  |
| --- | --- | --- |
| *lastSuccessfulBuild* | *text* | Ссылка на последнюю сборку |
| *splashScreenBackgroundColor* | *text* | Фон проекта |
| *font* | *text* | Шрифт |
| *languages* | *text* | Языки |
| *statusBarColor* | *text* | Цвет статус-бара |
| *mainColor* | *text* | Главный цвет |
| *accentColor* | *text* | Дополнительный цвет |
| *userId* | *text* | Индефикатор пользователя |

Таблица «Controls» отвечает за хранение информации о созданных элементах управления, находящихся на странице, она содержит: идентификатор элемента, название элемента, индефикатор страницы, свойства элемента, свойста элемента для генерации, размер, координаты элемента, тип родительского элемента, идентификатор родительского элемента, рассположение по горизонтали, длина элемента, рассположение по вертикали, высота элемента, стили элемента, тип элемента, идентификатор проекта, текст внутри элемента. Описание ее полей приведено в таблице 3.3.

Таблица 3.3 – Структура таблицы «Controls»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *text* | Идентификатор элемента |
| *name* | *text* | Название элемента |
| *pageId* | *text* | Индефикатор страницы |
| *properties* | *text* | Свойства элемента |
| *xamarinProperties* | *text* | Свойста элемента для генерации |
| *size* | *text* | размер |
| *coordinates* | *text* | Координаты элемента |
| *parentType* | *text* | Тип родительского элемента |
| *parentId* | *text* | Идентификатор родительского элемента |
| *gridColumn* | *text* | Рассположение по горизонтали |
| *gridColumnSpan* | *text* | Длина элемента |
| *gridRow* | *text* | Рассположение по вертикали |
| *gridRowSpan* | *text* | Высота элемента |
| *styles* | *text* | Стили элемента |
| *type* | *text* | Тип элемента |
| *projectId* | *text* | Идентификатор проекта |
| *text* | *text* | Текст внутри элемента |

Таблица «Users» содержит полную информацию о пользователе. Структура представлена в таблице 3.4.

Таблица 3.4 – Структура таблицы «Users»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *text* | Идентификатор пользователя |
| *name* | *text* | Имя пользователя |
| *login* | *text* | Логин |
| *passwordHash* | *text* | Хеш-версия пароля |
| *passwordSalt* | *text* | Данные для хеширования пароля |
| *userRole* | *integer* | Роль |
| *Email* | *text* | Электронная почта |

Таблица «ResetPassword» содержит информацию необходимую для восстановления пароля пользователем. Структура представлена в таблице 3.5.

Таблица 3.5 – Структура таблицы «ResetPassword»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *text* | Идентификатор пользователя |
| *requestTime* | *date* | Имя пользователя |
| *userId* | *text* | Логин |
| *userEmail* | *text* | Хеш-версия пароля |
| *resetPasswordHash* | *text* | Данные для хеширования пароля |

Выше были представлены таблицы, которые были созданы для функционирования всей системы веб-ресурса.

## Подключение к базе данных

Чтобы взаимодействовать с базой данных нужен контекст данных. Причем *Entity Framework Core* использует подход *Code First*, при котором нам надо сначала определить модели и контекст данных, а потом уже исходя и этих моделей и класса контекста будет создаваться база данных и все ее таблицы.

Чтобы создать контекст, нам надо унаследовать новый класс от класса *DbContext*. Свойства наподобие *public DbSet<Page> Pages { get; set; }* помогают получать из БД набор данных определенного типа (например, набор объектов *Page*).

Фактически каждое свойство *DbSet* будет соотноситься с отдельной таблицей в базе данных. По умолчанию у нас база данных отсутствуют. Поэтому в классе *Startup* определен вызов *Database.EnsureCreated()*, который при отсутствии базы данных автоматически создает ее. Если база данных уже есть, то ничего не происходит.

Следующим шагом в настройке проекта является изменение файла *Startup*.*cs*. В нем нам надо изменить метод *ConfigureServices*().

Контекст модели данных приложения отображен на рисунке 3.4.

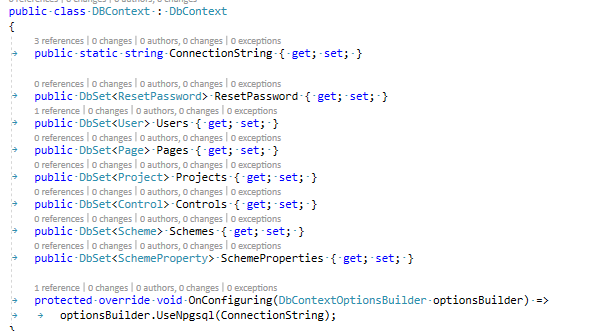


Рисунок 3.4 – Класс контекста базы данных

Чтобы подключаться к базе данных, нам надо задать параметры подключения. Параметры хранятся в файле *appsettings.json*. По умолчанию он содержит только настройки логгирования. Строку подключения можно увидеть на рисунке 3.5.



Рисунок 3.5 – Строка подключения к базе данных

Для получения доступа к базе данных создается экземпляр класса SimpleRepository при создании которого указывается строка соединения, за это отвечает код, представленный в листинге 3.2.

DBContext.ConnectionString = Config.GetConnectionString();

SimpleRepository.SetDbContext(typeof(DBContext));

Листинг 3.1 – Класс работы с базой данных

## Страница авторизации

Знакомство пользователя с приложением начинается со страницы авторизации и регистрации. Для того, чтоб пользователь смог войти в систему был разработан контроллер *AuthController* который содержит методы для входа в систему и регистрацию. Код этого контроллера можно увидеть на рисунке 3.6

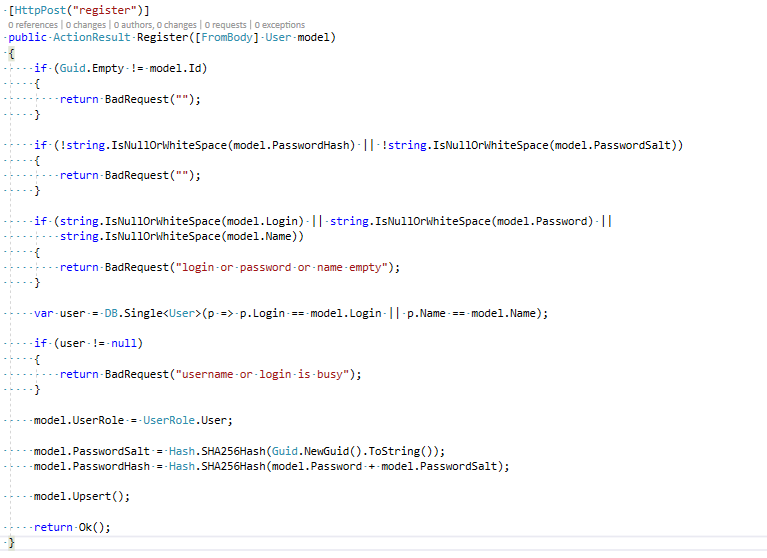


Рисунок 3.6 – строка подключения к базе данных

При попытке регистрации вначале идет проверка правильности заполнения данных. Механизм регистрации со стороны клиента отображен на рисунке 3.7



Рисунок 3.7 – Функция для регистрация со стороны клиента

Если какое-то из полей не заполнено, то сервер возвращает ответ с ошибкой, затем вызывается механизм хеширования пароля, так как сам пароль хранится в базе данных не может, ввиду правил безопасности, затем пользователь добавляется в базу данных и сервер возращает ответ клиенту, что пользователь добавлен, а на клиенте происходит переадресация на страницу авторизации.

Механизм авторизации строится на *cookie*-файлах. При попытке входа хешируется введенный пользователем пароль со сохранненой в базе данных пользовательской информацией. Если в базе данных есть такой пользователь и хеш введеного пароля совпадает с тем, что находится в базе, то пользователю отправляется его авторизационный токен, который потом будет проверятся при переходах между страницами. Метод авторизации со стороны сервера можно увидеть на рисунке 3.8



Рисунок 3.8 – Метод авторизации пользователя

После упешной аторизации пользователя перенаправляют на страницу списка проектов.

Так же к механизму авторизации можем отнести метод, который вызывается при каждом переходе между страницами и проверяет авторизационный токен, и если он отсутствует или неверен, то пользователя снова перенаправляют на страницу входа в приложения. Метод проверки токена основан на аттрибутах и middleware. Так написав одну функцию и наследовав ее от класса *Attribute* и интерфейса *IactionFilter* этот метод будет вызываться при каждой загрузке страницы. Реализацию этого метода можно увидеть на рисунке 3.9.



Рисунок 3.9 – Метод проверки авторизации

Весь механизм авторизации и регистрации работает как одно целое и позволяет избежать воздействия на систему из вне.

## Страница списка проектов

После успешной авторизации пользователь попадает на страницу списка проектов. Увидеть его реализацию можно на рисунке 3.10

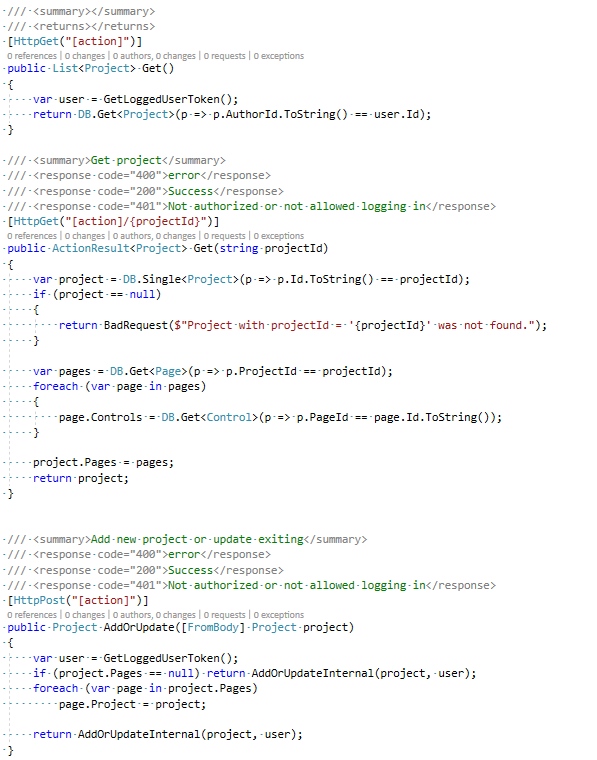


Рисунок 3.10 – Реализация ProjectController

Здесь рализовано добавление нового проекта, изменение существующего и удаление проктов. Логика отображения проектов находится на стороне клиента.

Блок-схема функции добавления нового проекта в базу данных представлена в приложении Г, а так же на рисунке 3.11.

|  |
| --- |
|  |

Рисунок 3.11 – Блок-схема функции добавления нового проекта в базу данных

После того как сервер вернул ответ, на страницу добавляются уже созданные проекты, на кнопки добавлются обработчики нажатий. Работа метода по добавлению показана на рисунке 3.12. Суть его работы заключается в получении списка всех проектов, со всеми необходимыми настройками. Затем вызывается функция-обработчик, которая достает нужные настройки проекта, и поэтапно начинает их отрисовку. Отрисовка состоит из нескольких этапов:

* добавление *css* свойств элементу;
* вставка динамических данных;
* создание обработчиков нажатия;
* добавление обработчиков на элемент;
* отрисовка созданного элемента на странице;
* завершение загрузки проектов.

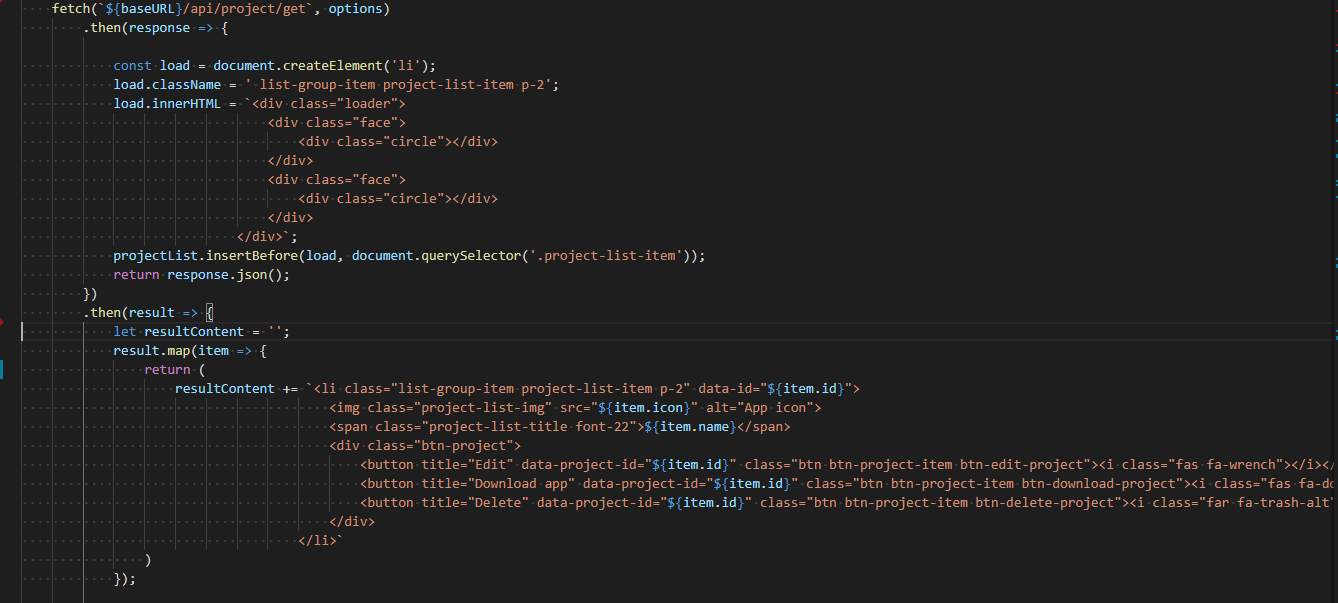


Рисунок 3.12 – Реализация добавления проектов на страницу

Для отправке запросов на сервер используется интерфейс *JavaScript* для работы с запросами и ответами *HTTP* - глобальный метод fetch(), который позволяет легко и логично получать ресурсы по сети асинхронно.

Подобная функциональность ранее достигалась с помощью *XMLHttpRequest*. *Fetch* представляет собой лучшую альтернативу, которая может быть легко использована другими технологиями, такими как *Service Workers*. *Fetch* также обеспечивает единое логическое место для определения других связанных с *HTTP* понятий, такие как *CORS* и расширения для *HTTP*. Поэтому во всем проекте для отправки *HTTP* запросов исользуется именно *Fetch*.

## Страница редактирования проекта

Для взаимодействия с клиенсткой частью страницы редактирования проекта есть три контроллера: *ProjectController*, *ControllController*, *PagesControllers*. Данные контроллеры разработаны аналогичным образом: в них при помощи *EntityFramework* методы получают определенную выборку данных, изменяют существующие данные, добавляют новые, и производят. Блок схема изменения проекта представлена в приложении А и на рисунке 3.13.

Логика изменения проекта состоит из нескольких ключевых мометов:

* получение списка проектов;
* проверка на то, что количество полученных проектов больше нуля;
* затем в зависимости от количества созданных проектов, происходит либо открытие существующего проекта, либо открывается страница создания нового;
* выгрузка проекта;
* внесение изменений в проект;
* сохранение изменений в БД;
* возвращение на страницу списка проектов.

|  |
| --- |
|  |

Рисунок 3.13 – Блок-схема алгоритма изменения проекта

За взаимодействие с самим проектом, например, изменение названия, фоновой картинки отвечает *ProjectController*.

За взаимодействие со страницами внутри проекта отвечает контроллер *PagesController*. *ControllController* отвечают за работу элементов управления на странице. Все контроллеры наследуются от одного контроллера «BaseController», в котором находится общая логика для всех контроллеров в проекте. Он является центральным звеном в архитектуре *ASP.NET Core MVC*. При получении запроса система маршрутизации выбирает для обработки запроса нужный контроллер и передает ему данные запроса. Контроллер обрабатывает эти данные и посылает обратно результат обработки. *В ASP.NET Core MVC* контроллер представляет обычный класс на языке *C#,* который наследуется от абстрактного базового класса *Microsoft.AspNetCore.Mvc.Controller*. Контроллер, как и любой класс на языке *C#*, может иметь поля, свойства, методы.

По умолчанию *BaseController* имеет четыре метода, которые можно назвать действиями. Действия контроллера - это публичные методы, которые могут сопоставляться с запросами. Например, стандартный контроллер содержит метод *Index* - он имеет модификатор *public* и поэтому может использоваться для обработки запроса.

Система привязки *MVC* сопоставляет параметры запроса и параметры метода по имени. То есть, если в строке запроса идет параметр a, то его значение будет передаваться именно параметру метода, который также называется a. При этом должно быть также соответствие по типу, то есть если параметр метода принимает числовое значение, то и через строку запроса надо передавать для этого параметра число, а не строку.

Реализация *ProjectController* представлена на рисунке 3.14.

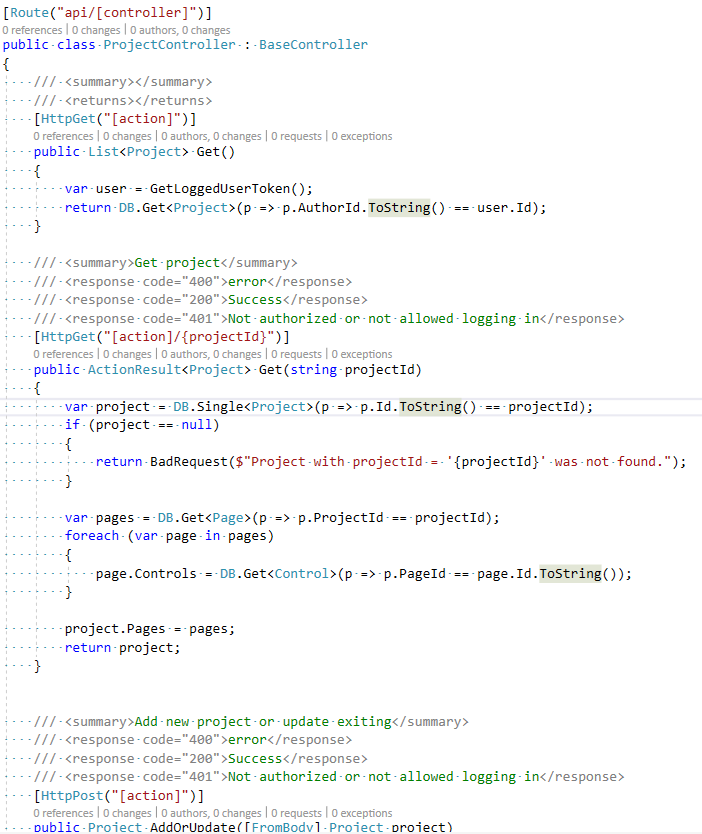


Рисунок 3.14 – Реализация контроллера ProjectController

На клиентской стороне при открытии страницы отправляется запрос на сервер, который возвращает клиенту список страниц, элементов и основных настроек проекта.Затем метод ищет главную страницу из списка все страниц и начинает ее отрисовку. Отрисовка элементов состоит из трех этапов: сразу отрисовываются статические части страцы, после получается список элементов для нужной страницы, а затем элементы с помощью *javascript* функции добавляются на страницу. Реализация этого метода отображена на рисунке 3.15.

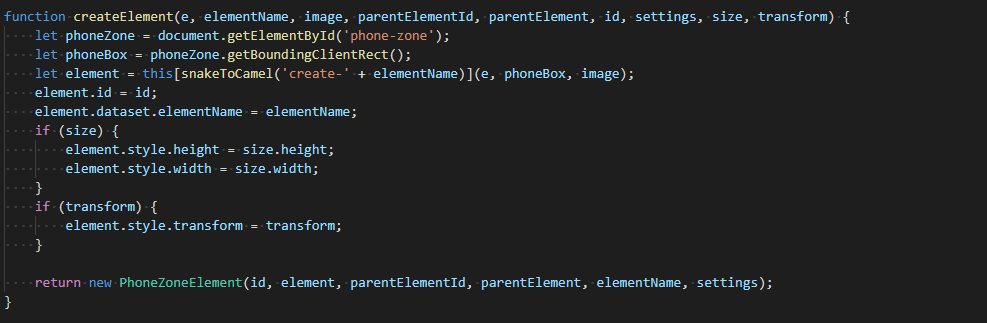


Рисунок 3.15 – Реализация метода createElement

Затем созданным элементам присваиваются ранее сохраненные свойства. Для того чтобы заполнить страницу статическими элементами, необходимо в левую панель элементов отрисовать нужные элементы, затем во второй вкладке нужно отобразить список страниц. Чтобы отрисовать все элементы вызывается функция *createElement*. Блок-схема алгоритма добавления элемента на страницу реактирования представлена в приложении В и на рисунке 3.16.

|  |
| --- |
|  |

Рисунок 3.16 – Блок-схема алгоритма добавления элемента на страницу

Алгоритм ее работы представлен несколькими основными действиями:

* выбор нужного элемента;
* выборка данных из БД;
* отрисовка элемента;
* сохранение новых данных в БД;
* редактирование элемента пользователем.

Так как каждый имеет свое параметры, которые позже могут быть изменены пользователем, то необходимо их описать при создании. Для этого, при создании вызывается еще одна функция, которая совпадает названием с типом элемента.

В ней происходит добавление всех своиств и параметров, присущих этому элементу, и которые позднее могут быть изменены пользователем.

Пример реализации метода добавления свойств отображен на рисунке 3.17.



Рисунок 3.17 – Реализация добавления свойств элементам

Добавление свойств элементу происходит с помощью функционала языка программирования *Javascript*.

Как правило, существует два способа задания стилей для элемента с помощью Javascript:

* создать класс в CSS и использовать его;
* писать стили непосредственно в атрибуте style.

Был выбран первый вариант, так как он помогает избавиться от написания лишнего кода и ускоряет процесс разработки, а так же это позволяет ускорить работу приложения. Ведь для каждого элемента будет использоваться уже готовый класс. За добавление списка страниц отвечает метод addPagesToList. Реализацию которого можно увидеть на рисунке 3.18



Рисунок 3.18 – Реализация добавления списка страниц

Принцип его работы заключается в следующем: клиент отправляет запрос к серверу, чтобы получить список всех страниц. Реализация метода-обработчика представлена на рисунке 3.19.



Рисунок 3.19 – Реализация функции обработки нажатия на страницу

Если по какой-то причине сервер возвращает сообщение об ошибке, то метод обработает его, и вернет пользователя на страницу со списком проектов. После того как сервер вернул список страниц, метод ищет главную страницу, так она должна быть в самом вверху, а так же быть помеченной отдельным символов. Затем запускается цикл, который создает кнопки, в которых содержится название страницы и обработчик нажатия на нее. Так как при нажатии по какой-нибудь странице происходит ее отрисовка. В приложении было реализовано добавление двух типов страниц: обычная и всплывающая. Различие между ними заключается в том, что всплывающая страница всегда находится поверх обычной, и не может существовать самостоятельно. При добавлении новой страницы, вызвается функция которая в зависимости от типа страницы добавляет ей нужные свойства. Реализацию этой функции можно увидеть на рисунке 3.20

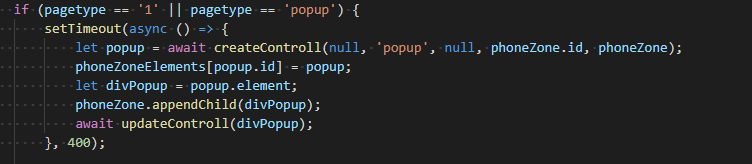


Рисунок 3.20 – Реализация добавления свойств страницы

Реализация отрисовки всплывающей страницы отличается от отрисовки обычной страницы. Функция отвечающая за ее отрисовку отображена на рисунке 3.21

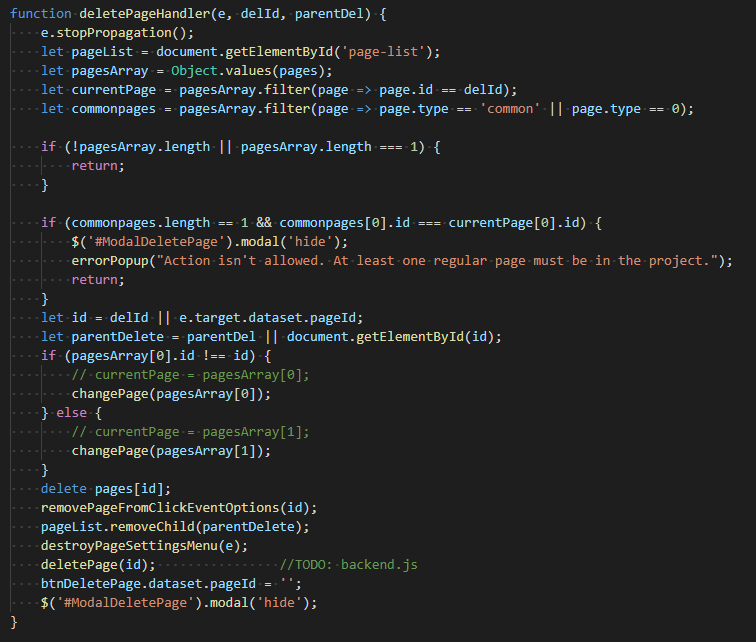


Рисунок 3.21 – Реализация функции отрисовки сраниц

Для того, чтобы было легко найти нужную страницу из списка всех страниц, был реализован метод поиска нужной страницы по ее названию, его реализация представлена на рисунке 3.22

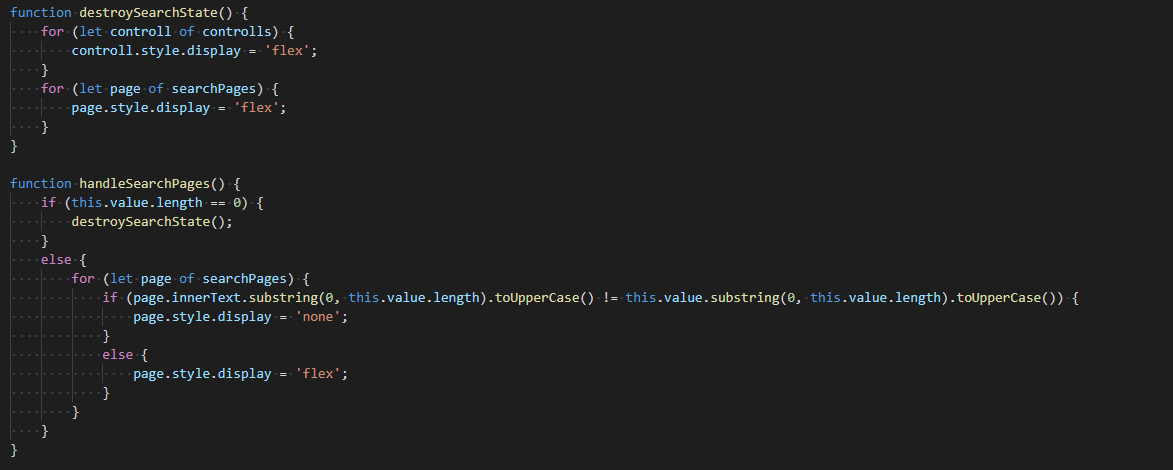


Рисунок 3.22 – Реализация функции поиска страницы

Суть его работы заключается в том, что проверяется наличие введенных пользователем элемементов среди всего списка страниц, и если есть совпадения, то список страниц очищается и в него добавляются нужные страницы.

После выбора страницу вызывается функция которая отвечает за восстановления всех элементов, которые были добавленны ранее на эту страницу. Реализацию этой функции можно увидеть на рисунке 3.23

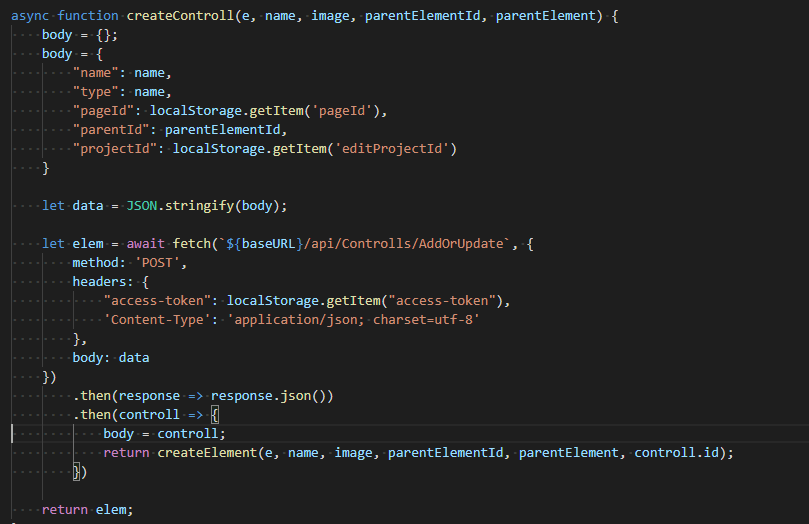


Рисунок 3.23 – Реализация функции восстановления элементов

После восстановления элементов, необходимо заставить их двигаться.

Для этого вызывается функция *createMoveableInstance*. Суть работы которой заключается в:

* определение границ контейнера в котором может двигаться элемент;
* определение размеров и центра элемента;
* добавление рамки вокруг элемента, для изменения его размера;
* добавление обработчиков событий движения.

Так же в этой функции задаются ограничения для движения некоторых элементов. Пример реализации представлен на рисунке 3.24.

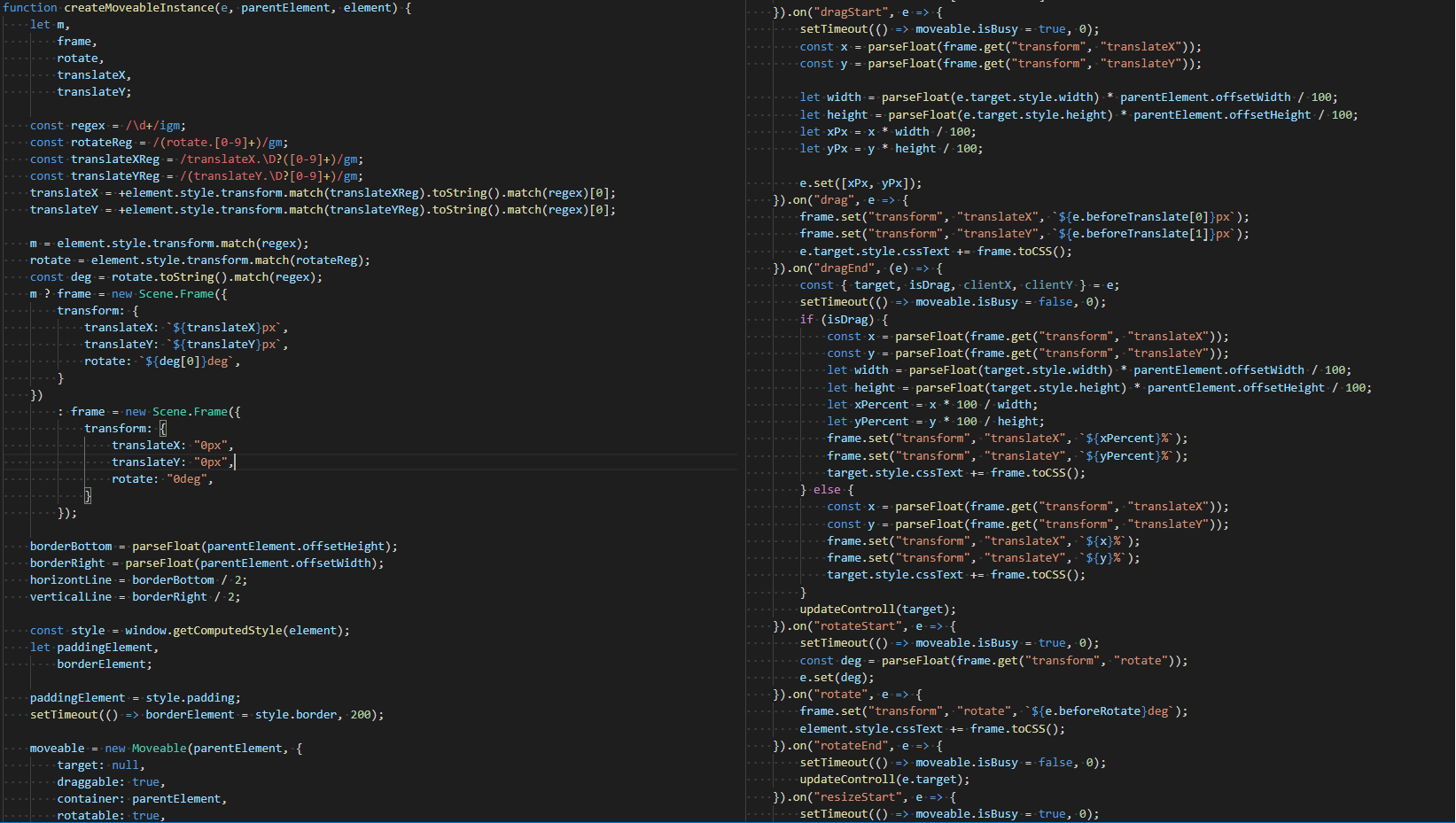


Рисунок 3.24 – Реализация функции движения элементов

Для настройки созданных элементов используется боковая панель. Пример реализации настроек для одного элемента можно увидеть на рисунке 3.25.



Рисунок 3.25 – Пример реализации настроек элемента

При клике по элементу вызывается несколько функций. Первая очищает все, что есть на панели настроек. Вторая считывает возможные настройки элементов и добавляет нужные элементу управления и одновременно добавляет на них обработчики.

После того как в настройки элемента вносятся изменения, отправляется запрос на сервер, который обновляет значение в базе данных. За счет этого достигается автосохранение проекта и нет необходимости в ручном сохранении проекта.

## Режим предпросмотра

Для того, чтобы пользователю было удобнее разробатывать интерактивный интерфейс, была разработана возможность предпросмотра созданного прототипа интерфейса. С технической точки зрения, это обычная *HTML* страница, которая выглядит как мобильное устройста. Принцип ее работы схож со страницей редактирования проекта. При нажатии на кнопку предпросмотра, вызывается функция, которая начинает отрисовку страницы. Вначале делается запрос на сервер и получается главная страница, так как с нее и начинается запуск прототипа интерфейса. Реализацию этой функции можно увидеть на рисунке 3.26

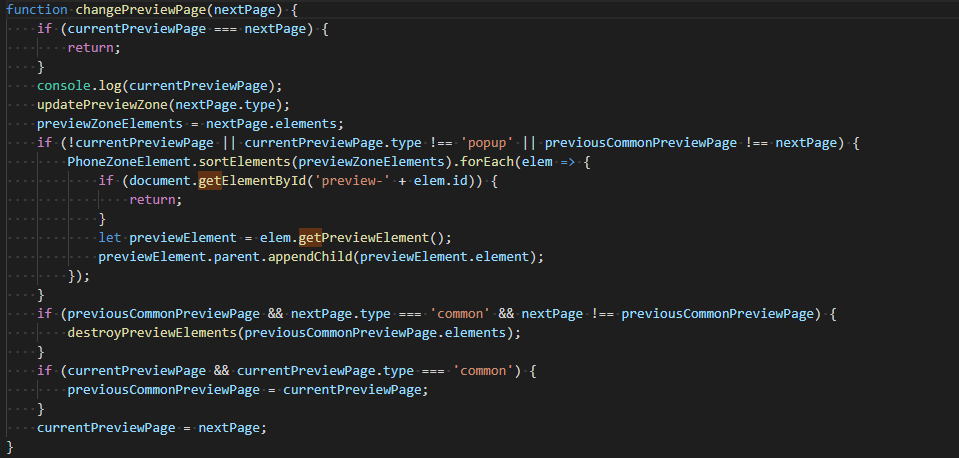


Рисунок 3.26 – Реализация получения главной страницы

После того, как главная страница была получена, теперь необходимо добавить на нее все элементы, а так же необходимо сделать их интерактивным, это значит, что если в редакторе на какой-то элемент добавили переход между страницами, то и в режиме предпросмотра он тоже должен работать.

## Генератор мобильного приложения

Для того, чтобы сгенирировать мобильное приложение, клиент отправляет запрос на сервер с индефикатором проекта, для которого нужно собрать мобильное приложение. Затем вызывается мотод *GenerateApk* в контроллере *GeneratorController*, который собирает *json* файл со всеми настройками проекта, в которе входят список страниц и всех элементов, которые туда входят. Однако этот файл имеет неверную структуру для генерации. Поэтому этот json фаил нужно конвертировать в в фаил с такой структурой, которая будет понятна генератору. Блок схема алгоритма генерации мобильного приложения представлена в приложении Б и на рисунке 3.27.

|  |
| --- |
|  |

Рисунок 3.27 – Блок схема алгоритма генерации мобильного приложения

Работа алгоритма состоит из нескольких основных шагов:

* получение данных из бд;
* создание пустого проекта мобильного приложения;
* выбор из бд списка элементов для каждой страницы;
* вставка элементов в страницу проекта;
* заполнение проекта основными настройками;
* сборка проекта;
* цифровая подпись собранного проекта;
* вывод пути к собранному приложению;
* скачивание готового приложения.

Для этого был разработан класс конвертер *Convert*, который является связующим звеном между веб приложением и генератором и вынесен отдельным проектом, для того, чтобы в дальнейшем его можно было расширить другими платформами.

Суть работы конвертера заключается в сопоставлении моделе данных, которые есть в веб приложении и моделей поддерживаемых генератором, так как они не совпадают из-за особености операционной системы *android*. Реализацию метода конвертации можно увидеть на рисунке 3.28

|  |
| --- |
|  |

Рисунок 3.28 – Реализации конвертера json

После функции конвертации на выход получаем *json* фаил, который может быть обработан генератором. Для отправки json файла генератору создается объект класса *Disigner*, который начинает сборку проекта.

Генератор создает пустой Xamarin.Forms проект. В него он добавляет подготовленные заранее шаблонные файлы. Затем, генератор начинает разбор *json* на составные части.

Вначале генератор находит главную страницу проекта прототипа, затем задает основные настройки проекта, такие как название, иконка и тд.

Пример реализации можно увидеть на рисунке 3.29 и в приложении Е.

|  |
| --- |
|  |

Рисунок 3.29 – Реализации генерации основных настроек проекта

После того как была сгенерированна главная страница, начинается генерация всех остальных страниц. Так как все страницы связаны между собой, их генерация проводится в определенном порядке. Сначала создаются родительские страницы, затем дочерние.

Когда все страницы созданы, начинается сборка проекта. Генератор считывает фаил настроек, в котором хранится имя папки, в которую будет собран *apk* фаил, ключи для подписи приложения, и путь для выведения лог-файла.

Сборка проекта происходит с помощью *MsBuild* - платформа сборки проекта для .*Net* платформы.

Реализация метода сборки отображена на рисунке 3.30

|  |
| --- |
|  |

Рисунок 3.30 – Реализации сборки проекта

После того как сборка успешно завершена, генератор возвращает ответ, который содержит путь к сгенерированному *apk*-файлу, А веб-приложение начинает скачивание. Пользователь получает полностью готовое к установке мобильное приложение, которое и будет являтся итерактивным прототипом интерфейса.

## Вывод по разделу

В результате выполнения дипломного проекта был разработано веб-приложение генератор интерактивных прототипов интерфейсов для операционной системы *android* в отличии от аналогов, проект имеет простой и понятный каждому интерфейс, протипы генерируются в виде apk файлов, что позволяет удобно обмениваться прототипом между другими участниками команды и отчетливо увидеть, как прототип смотрится на разных мобильных устройствах с разным разришением экрана.

Полученное веб-приложение не требует специфического программного обеспечения, для его запуска необходимо наличие установленного интернет-браузера на компьютере пользователя, доступ в сеть *Internet* любым доступным и удобным для пользователя способом, а также доступ к имеющимся данным. Проект является интуитивно понятным и простым. В программном средстве использованы устойчивые формы записи кода (использование обработчиков ошибок) благодаря чему повышается отказоустойчивость приложения.

Веб-ресурс справляется со всеми поставленными задачами и целями, а именно:

* спроектирована и разработана структура базы данных;
* разработан интерфейс авторизации, управления проектами, а так же создания и генерации прототипа;
* реализована генерация готового к установке мобильного приложения;
* реализована полная настройка существующего проекта и всех основных элементов добавленных на страницу.

Спроектированная архитектура проекта, позволит добавлять и усовершенствовать функциональные возможности и поддерживать работу системы в будущем.

# Тестирование программного средства

ФИО

Подпись

Дата

Лист

1

БГТУ 04.00.ПЗ

Разраб.

*Вольский Д.М.*

Пров.

Жиляк Н.А.

Консульт.

Жиляк Н.А.

Н. контр.

Рыжанкова А.С.

Утв.

Смелов В.В.

4 Тестирование программного средства

Лит.

Листов

5

*74218005, 2020*

Тестирование является неотъемлемой частью процесса разработки программного продукта, так как данный этап позволяет найти неисправности в работе приложения и исправить их.

В данной главе будут описаны некоторые тесты, проведенные для выявления и устранения неисправностей разработанного программного продукта, а так же описаны механизмы тестирования продукта во время его разработки.

Цели тестирования могут отличаться, в зависимости от этапа разработки ПО, на котором оно проводится. К примеру, на этапе кодирования целью тестирования будет вызов как можно большего количества сбоев в работе программы, что позволит локализовать и исправить дефекты. В то же время, при приемочном тестировании необходимо показать, что система работает правильно. В период сопровождения, тестирование в основном необходимо для того, чтобы убедится в отсутствии новых багов, появившихся во время внесения изменений. Главная же задача тестирования – поиск дефектов.

Для начала представим названия тест-кейсов, которые и будем рассматривать ниже:

– создание пользователя;

– создание проекта;

– удаление проекта;

– редактирование проекта.

## Тестирование продукта с помощью unit тестов

*Unit* тестирование -  процесс , позволяющий проверить на корректность отдельные модули [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки. Для юнит тестирования был выбран модуль, отвечающий за конвертацию *json* файла в нужный формат. Так как этот модуль отвечает за связь с генератором, и если на этапе конвертации появятся ошибки, то приложение не будет сгенерированно.

Для тестирования этого модуля был создан еще один проект, который включает в себя два теста. При запуске этого теста теста, конвертер пытается обработать *json* фаил, который изначально неверный. Что позволит исключить ситуации, когда конвертер обрабывает неверный фаил, и соответсвенно отдает неверный ответ. Второй тест напротив, пытается обработать правильный фаил, и должен показать то, что на выходе фаил имеет правильную структуру и генератор сможет его обработать.

Реализацию unit тестов можно увидеть на рисунке 4.1

|  |
| --- |
|  |

Рисунок 4.1 – Реализации unit тестов

## Тестирование создания пользователя

В данном тесте будет проверена правильность ввода данных, а точнее их валидация до момента отправки данных на сервер.Для этого введем, например, в поле с электронной почтой не верное значение. Результат можно увидеть на рисунке 4.2.

|  |
| --- |
|  |

Рисунок 4.2 – Проверка входа с некорректной почтой

После ввода некорректной электронной почты отобразилось сообщение об ошибке, говорящее о том, что необходимо ввести корректные данные.

## Тестирование создания проекта

При добавлении нового проекта он должен добавится с указанным нами именем и загруженной картинкой, а так же отобразится в списке проектов.

На рисунке 4.3 показано добавление нового проекта

|  |
| --- |
|  |

Рисунок 4.3 – Добавление нового проекта прототипа интерфейса

После добавления проекта выходим на страницу со списком проектов и проверяем результат, который отображен на рисунке 4.4

|  |
| --- |
|  |

Рисунок 4.4 – Добавление нового проекта прототипа интерфейса

Как видно на рисунке, проект появился в списке с нужным именем.

## Тестирование удаления проекта

После создания проекта нужно убедиться, что метод удаления работает корректно. Для это мы попробуем удалить ранее созданные проект. На рисунке 4.5 видно, что система вначале спросит, действительно ли мы хотим его удалить.

|  |
| --- |
|  |

Рисунок 4.5 – Попытка удаления проекта

После нажатия на клавишу удалить, проект пропадает из списка, как видно на рисунке 4.6.

|  |
| --- |
|  |

Рисунок 4.6 – Список проектов

После перезагрузки страницы, проект больше не отображается в списке, а значит он полностью удален.

## Тестирование редактирования проекта

Для того чтобы проверить, что редактирование проекта работает корректно, созданим еще один проект. Зададим ему имя и картинку которые показаны на рисунке 4.7

|  |
| --- |
|  |

Рисунок 4.7 – Созданный проект

Затем изменим у проекта имя и картинку и вернемся снова на страницу списка проектов. Результат представлен на рисунке 4.8

|  |
| --- |
|  |

Рисунок 4.8 – Отредактированный проект

Как видно на изображении, проект был успешно отредактирован и изменен.

## Выводы по разделу

В данной главе было выполнено несколько контрольных примеров для обнаружения ошибок в поведении системы и выполнен ряд тестов для проверки ее стабильности. В результате проделанной работы установлено что система стабильно работает в различных ситуациях и является стрессоустойчивой.

# Руководство пользователя

ФИО

Подпись

Дата

Лист

1

БГТУ 05.00.ПЗ

Разраб.

*Вольский Д.М.*

Пров.

Жиляк Н.А.

Консульт.

Жиляк Н.А.

Н. контр.

Рыжанкова А.С.

Утв.

Смелов В.В.

5 Руководство пользователя

Лит.

Листов

6

*74218005, 2020*

## 5.1 Авторизация пользователя

Для проверки авторизации пользователя используются логин и пароль. Форма авторизации представлена на рисунке 5.1.

|  |
| --- |
|  |

Рисунок 5.1 – Фрагмент экрана с формой авторизации пользователя

При введении неверных данных, пользователь увидит уведомление с ошибкой (рисунок 5.2).

|  |
| --- |
|  |

Рисунок 5.2 – Фрагмент экрана с ошибкой в форме авторизации пользователя

Это поможет пользователю понять, что в системе пошло не так.

## 5.2 Регистрация пользователя

Для того чтобы создать новый аккаунт, пользователь может нажать на кнопку «Register» на странице входа, после чего он будет перенаправлен на страницу с формой регистрации, которая изображена на рисунке 5.3.

|  |
| --- |
|  |

Рисунок 5.3 – Форма регистрации нового пользователя

При заполнении полей формы, введенные данные проверяются с помощью механизма валидации и регулярных выражений, а также проверяется совпадения пароля и его подтверждения, в случае каких-либо несоответствий требованиям пользователь увидит сообщения с ошибками (рисунок 5.4).

|  |
| --- |
|  |

Рисунок 5.4 – Форма регистрации с ошибками

После успешного прохождения валидации и удачного прохождения регистрации, пользователь будет перенаправлен на страницу со списком проектов.

## Страница списка проектов

На данной странице отображены созданные пользователем проекты прототипов интерфейсов. Здесь он может выбрать нужный ему проект, изменить существующий, либо, удалить проект, который больше не нужен. Для большего удобства проекты отображаются в виде плиток с картинкой, чтобы было проще найти нужный проект. Реализацию страницы можно увидеть на рисунке 5.5

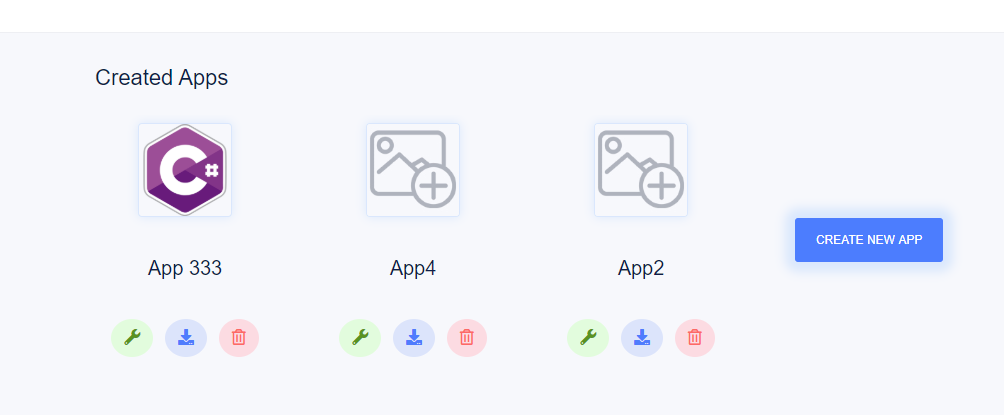


Рисунок 5.5 – Страница со списком проектов

При нажатии на кнопку создать новое приложение открывается страница редактирования проекта, где справа появляется меня с основными настройками такими как цвет, название, картинка приложения, загрузочная картинка. При внесении каких-либо изменений данные автоматически обновятся на сервере. Это избавляет пользователя от постоянного сохранения проекта. Добавление нового проекта можно увидеть на рисунке 5.6

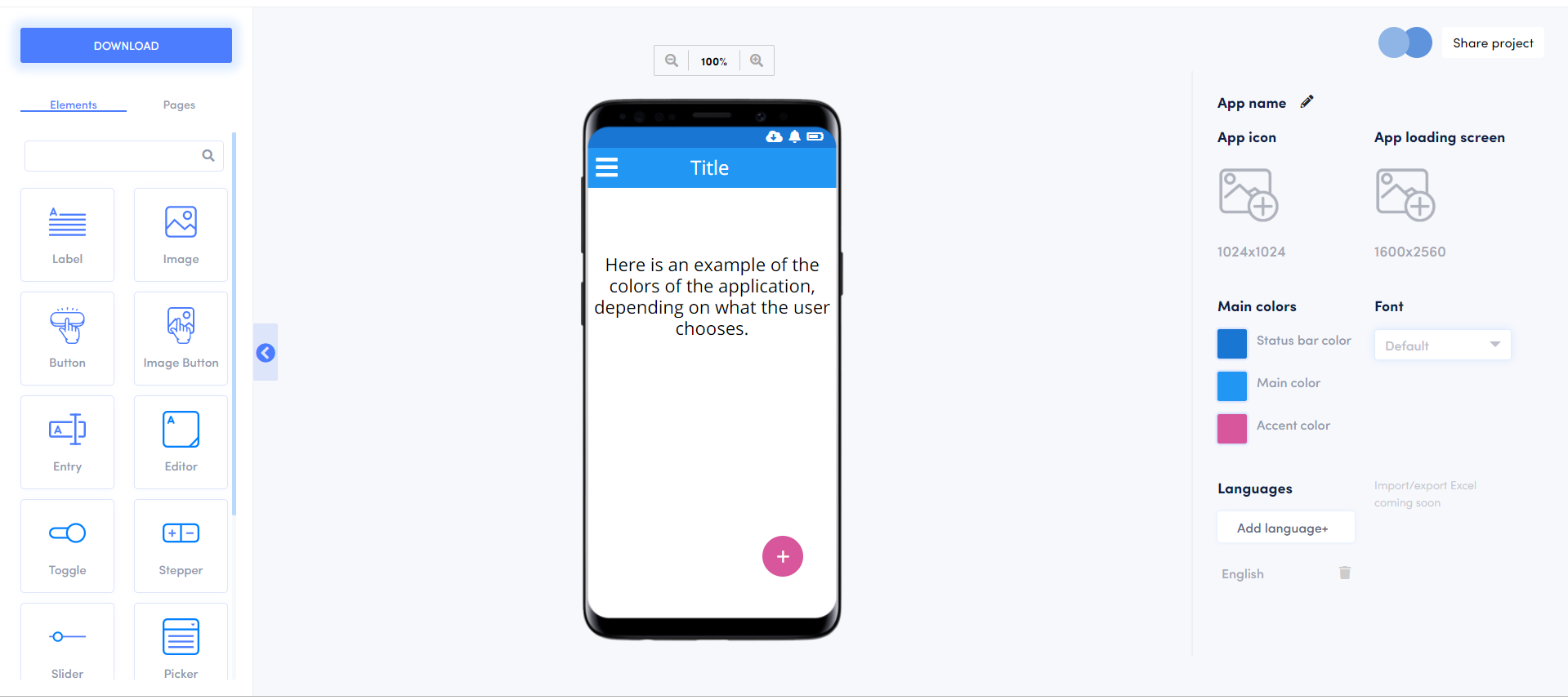


Рисунок 5.6 – Добавление нового проекта

После того, как проект создан, пользователь может начать редактировать.

## Страница редактирования проекта

В левой части экрана пользователь может выбрать любой элемент и перетащить его на рабочую область. После того, как элемент будет вытащен, у пользователя появляется возможность изменять его свойтва, а так же двигать. Добавление элемента на рабочую панель показано на рисунке 5.7

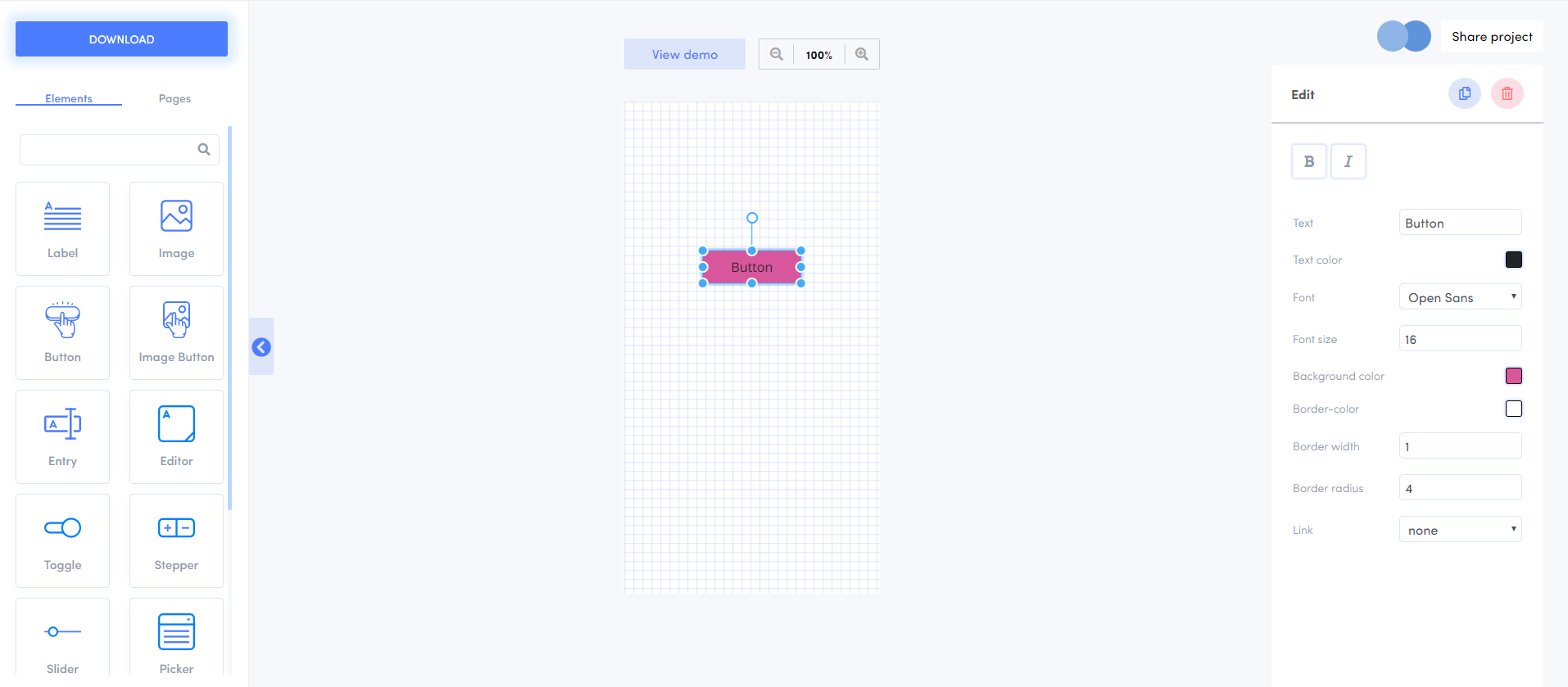


Рисунок 5.7 – Добавление нового элемента

Так же на рисунке 5.7 показана панель настройки элемента. Она находится в правой части экрана, и меняется в зависимости от типа элемента. Здесь так же реализован механизм автосохранения. То есть пользователю не нужно будет каждый раз сохранять свои изменения.

Так же для удобства пользователя на странице реализована функция увеличения маштаба рабочей области, которая показана на рисунке 5.8

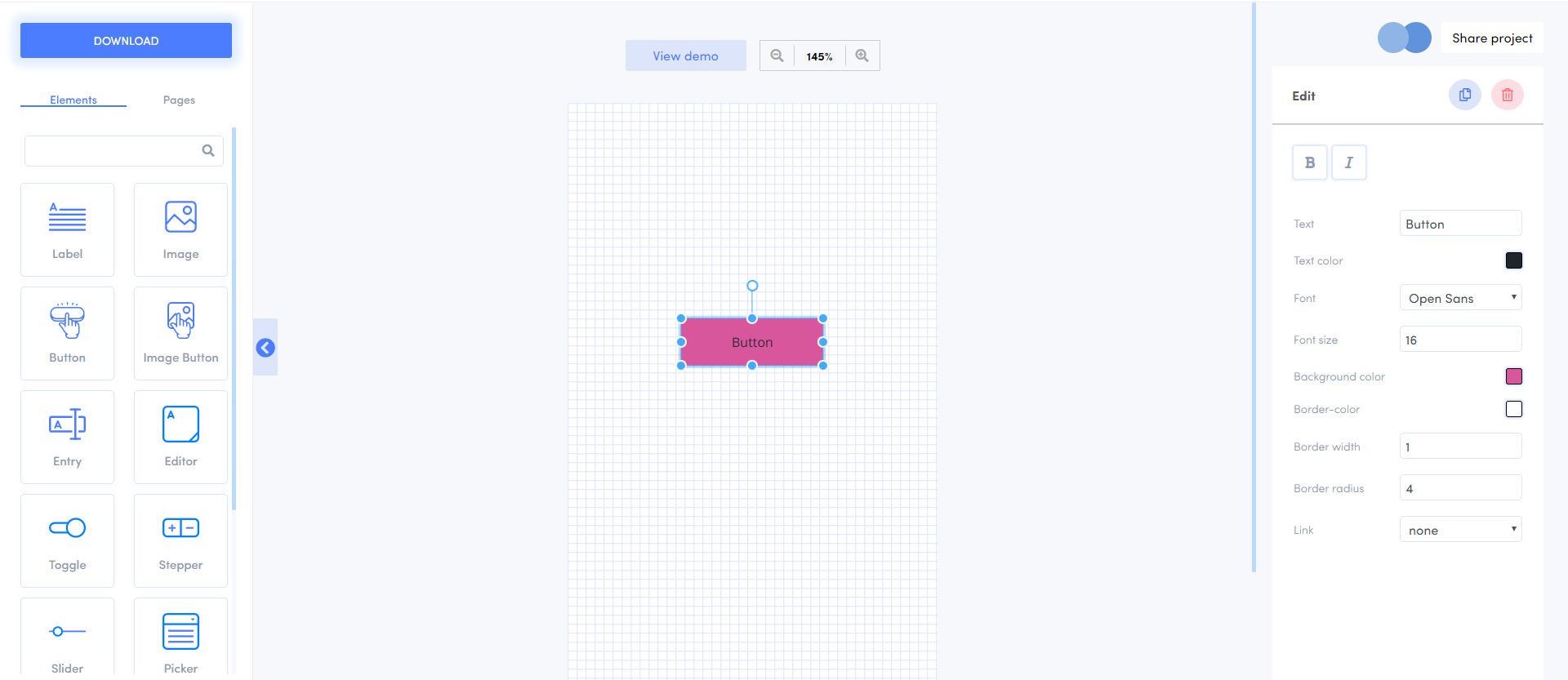


Рисунок 5.8 – Увеличение масштаба

На этом же экране пользователь может управлять своими страницами. Для этого в левом углу нужно нажать на вклатку Pages, где открывается список созданных страниц. Отсюда можно добавить новую страницу как показано на рисунке 5.9

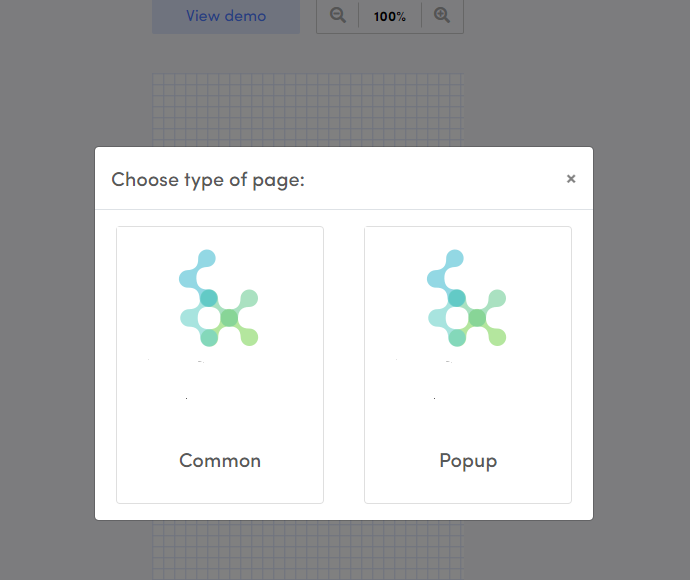


Рисунок 5.9 – Создание новой страницы

Как и у элементов, у страницы тоже есть свои свойства, которые изменяются.

После того, как пользователь внес нужные изменения, он может генерировать проект, но перед тем как начать генерацию, у него есть возможность посмотреть, как будет выглядеть его прототип интерфейса в браузере. Для этого нужно нажать на кнопку «view demo», после чего откроется всплывающее окно, в котором можно увидеть готовы прототип. Результат работы можно увидеть на рисунке 5.10.

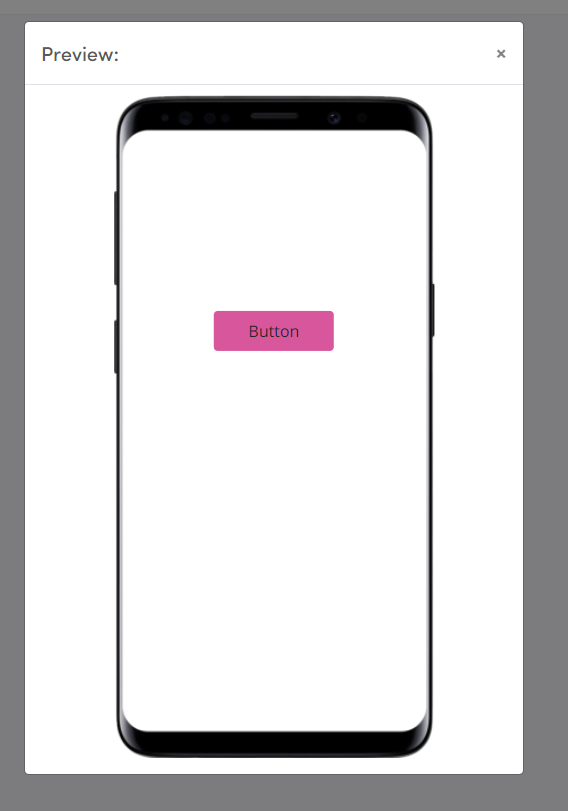


Рисунок 5.10 – Превью режим

Когда пользователя все устраивает, он может начать собирать свой интерактивный прототип интерфейса в мобильное приложение.

## Генерация мобильного приложения

Для генерации приложения, пользователю нужно нажать всего одну клавишу «download», после чего начинается сборка проекта. Весь процесс занимает около двух минут, в это время пользователь видит анимацию, которая говорит ему о том, что процесс еще не завершен. Увидить анимацию можно на рисунке 5.11

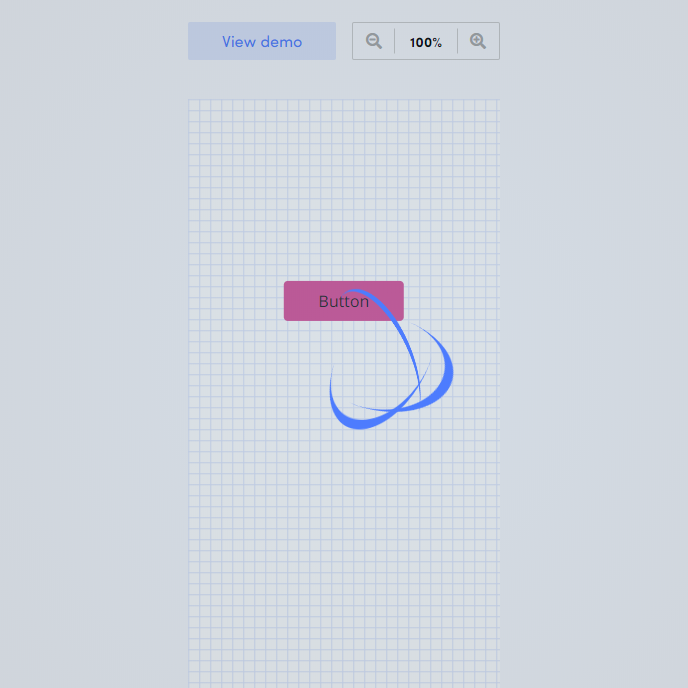


Рисунок 5.11 – Сборка мобильного приложения

После того, как процесс сборки проекта будет завершен, прложение с готовым интерактивным интерфейсом будет автоматически скачено на устройство пользователя.

5.6 Выводы по разделу

Данный раздел предоставляет пояснения по работе пользователя с веб-приложением. Так как интерфейс приложения прост и интуитивно понятен, то у пользователя не может возникнуть трудностей в его эксплуатации.

Архитектуру приложения и большие функциональные возможности, предоставляемые пользователю, получилось ограничить небольшим количеством экранов, что является прямым доказательством хорошо спроектированного и простого пользовательского опыта в данном приложении.

6 Технико-экономическое обоснование проекта

ФИО

Подпись

Дата

Лист

1

БГТУ 06.00.ПЗ

Разраб.

*Вольский Д.М.*

Пров.

Жиляк Н.А.

Консульт.

Евлаш А.И.

Н. контр.

Рыжанкова А.С.

Утв.

Смелов В.В.

6 Технико-экономическое обоснование проекта

Лит.

Листов

9

*74218005, 2020*

Главной целью экономического раздела является экономическое обоснование целесообразности разработки программного средства (ПС), представленного в дипломном проекте. В данном разделе пояснительной записки проводится расчет затрат на всех стадиях разработки и расчет экономии основных видов ресурсов в связи с использованием данного ПС.

Разработка проектов ПС требует разнообразных затрат и нередко значительных объемов ресурсов (трудовых, материальных, финансовых). В связи с этим, разработка и реализация каждого проекта должна быть обоснована, как технически, так и экономически. Программное обеспечение предполагает использование в коммерческих целях.

6.1 Общая характеристика разрабатываемого веб-приложения

Основной целью экономического раздела является экономическое обоснование целесообразности разработки веб-приложения, представленного в дипломном проекте.

В этом разделе пояснительной записки проводится расчет затрат на всех стадиях разработки, а также анализ экономического эффекта в связи с использованием данного веб-приложения.

Во время разработки дипломного проекта использовалась технология *ASP.NET Core MVC, Xamarin.Forms*, среды разработки *Visual Studio 2019*. Разработанное программное средство представляет собой генератор интерактивных интерфейсов для операционной системы android . Оно позволяет пользователю создать собственный прототип интерактивного интерфейса, который затем будет собран в мобильное приложение. Приложение является веб-ресурсом, благодаря чему может быть запущено и с персонального компьютера, и с мобильного устройства.

6.2 Исходные данные

Исходные данные для расчета стоимости программного продукта представлены в таблице 6.1.

Таблица 6.1 – Исходные данные расчета

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
| Численность разработчиков | чел. | Чр | 1 |
| Норматив дополнительной заработной платы | % | Ндз | 12 |

Окончание таблицы 6.1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
| Ставка отчислений в Фонд социальной защиты населения | % | Нфсзн | 34 |
| Ставка отчислений в БРУСП «Белгосстрах» | % | Нбгс | 0,4 |
| Цена одного машино-часа | руб. | Смч | 0,05 |
| Норматив прочих затрат | % | Нпз | 7,5 |
| Норматив накладных расходов | % | Нобп*,* обх | 110 |
| Норматив расходов на сопровождение и адаптацию | % | Нрса | 10 |
| Норма расхода материалов в расчете на сто строк исходного кода | руб. | Нм | 0,46 |
| Ставка НДС | % | Нндс | 20 |

В ходе проведения маркетингового анализа, была выявлена стоимость разработки системы генерации интерактивных интерфейсов. Средняя цена разработки аналогичного продукта составляет 45000-55000 рублей. Общая стоимость разработки программного средства, выбранного в качестве базы сравнения, составит 49000 руб.

6.3 Методика обоснования цены

В современных рыночных экономических условиях веб-приложение выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид веб-приложения, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки веб-приложения являются научно-технической продукцией. Широкое применение вычислительных технологий требует постоянного обновления и совершенствования веб-приложения. Выбор эффективных проектов веб-приложения связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя. У разработчика экономический эффект выступает в виде чистой прибыли от реализации веб-приложения, остающейся в распоряжении организации, а у пользователя в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
* ускорение ввода в эксплуатацию новых систем;
* улучшения показателей основной деятельности в результате использования веб-приложения.

Стоимостная оценка веб-приложения у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на оплату машинного времени;
* накладные расходы;
* прочие прямые затраты.

На основании затрат рассчитывается себестоимость и отпускная цена веб-приложения.

## 6.3.1 Объем веб-приложения

|  |  |  |
| --- | --- | --- |
|  |  | (6.1) |

Для оценки объема веб-приложения, все его функции классифицируются с использованием специального каталога функций, который определяет их объем. Общий объем веб-приложения Vo, вычисляется как сумма объемов Vi каждой из n его функций (формула 6.1).

где *Vi* – объем *i*-ой функции ПС, условных машинных команд;

*n* – общее число функций.

В таблице 6.2 представлены функции, присутствующие в рассматриваемом программном средстве и соответствующий им объем в условных машинокомандах.

Таблица 6.2 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| Номер функции | Содержание функции | Объем, условных машино-команд |
| 101 | Организация ввода информации | 230 |
| 102 | Контроль, предварительная обработка | 700 |
| 111 | Управление вводом/выводом | 1020 |
| 202 | Взаимодействие между компонентами системы | 840 |
| 401 | Взаимодействие с базой данных | 990 |
| 402 | Вспомогательные методы | 750 |
| 506 | Обработка ошибочных и сбойных ситуаций | 630 |
| 707 | Графический вывод результатов | 2700 |
|  | Итого | 8070 |

Опираясь на данные таблицы 6.2, можно определить объем веб-приложения разработанного в ходе дипломного проектирования:

*Vo* = 230 + 700 + 1020 + 840 + 990 + 750 + 630 + 2700 = 7860

(условных машино-команд).

Уточненный объем веб-приложения *Vo*/ равен произведению объема веб-приложения *Vo* на коэффициент изменения скорости обработки информации Кск (формула 6.2).

|  |  |  |
| --- | --- | --- |
|  | *Vo*/ = *Vo*· Кск | (6.2) |

Исходя из вычисленного объема веб-приложения, можно определить уточненный объем веб-приложения:

*Vo*/ = 7860 ⋅ 0,6 = 4716 (условных машино-команд).

## 6.3.2 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере веб-программирования на C#. В итоге было установлено, что средняя месячная заработная плата на позиции *junior/middle* составляет 1700 рублей.

Проект разрабатывался одним человеком на протяжении пяти месяцев. Таким образом, основная заработная плата будет рассчитываться по формуле (6.3):

|  |  |
| --- | --- |
| , | (6.3) |

где Соз – основная заработная плата, руб.;

Траз – время раработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата, руб.

Соз = 5 · 1 · 1700 = 8500 руб.

## 6.3.3 Дополнительная заработная плата

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (6.4):

|  |  |
| --- | --- |
|  | (6.4) |

где Соз – основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

Cдз = 8500 ⋅ 12 / 100 = 1020 руб.

## 6.3.4 Отчисления в Фонд социальной защиты населения

Отчисления в Фонд социальной защиты населения (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле (6.5):

|  |  |
| --- | --- |
|  | (6.5) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата за веб-приложение, млн. руб.;

Нфсзн– норматив отчислений в Фонд социальной защиты населения, %.

Cфсзн = (8500 + 1020) ⋅ 34 / 100 = 3236,2‬ (руб.).

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.6:

|  |  |
| --- | --- |
|  | (6.6) |

руб.

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 38,08 руб., а в фонд социальной защиты населения – 3236,2 руб.

## 6.3.5 Расходы на материалы

Сумма расходов на материалы СМ определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода НМ на уточненный объем веб-приложения Vo/ (формула 6.7).

|  |  |  |
| --- | --- | --- |
|  | . | (6.7) |

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равен 0,46 руб., можно определить сумму расходов на материалы:

СМ = 0,46 ⋅ 4716 / 100 = 21,69 руб.

Таким образом, сумма расходов на материалы составила 21,69 рублей.

## 6.3.6 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем веб-приложения Vo/ и на норматив расхода машинного времени на отладку ста строк исходного кода НМВ (формула 6.8).

|  |  |  |
| --- | --- | --- |
|  | . | (6.8) |

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен можно определить сумму расходов на оплату машинного времени:

Смв = 0,05 ⋅ 4716 ⋅ 15 / 100 = 35,37 руб.

Таким образом, расходы на оплату машинного времени составляют 35,37 рублей.

## 6.3.7 Прочие прямые затраты

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз (формула 6.9).

|  |  |  |
| --- | --- | --- |
|  | . | (6.9) |

Спз = 8500 ⋅ 7,5 / 100 = 637,5 руб.

Таким образом, прочие прямые затраты составили 637,5 рубля.

## 6.3.8 Накладные расходы

Сумма накладных расходов Cобп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп, обх (формула 6.10).

|  |  |  |
| --- | --- | --- |
|  | . | (6.10) |

где Снр – сумма накладных расходов, руб.;

Ннр – норматив накладных расходов в целом по организации, %;

Спз – сумма прочих затрат, руб.;

Соз – основная заработная плата, руб.

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

Собп, обх = 8500 ⋅ 110 / 100 = 9350 руб.

## 6.3.9 Сумма расходов на разработку веб-приложения

Сумма расходов на разработку веб-приложения Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов (формула 6.11).

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + См + Смв + Спз + Собп, обх | (6.11) |

где Ср – сумма расходов на разработку веб-приложения, руб.;

Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата на конкретное веб-приложения, руб.;

Сфсзн – сумма отчислений в Фонд социальной защиты населения, руб.;

См – сумма расходов на материалы, руб.;

Смв – сумма расходов на оплату машинного времени, руб.;

Спз – сумма прочих затрат, руб.;

Собп, обх – сумма накладных расходов, руб.

Ср = 8500 + 1020 + 3236,2 + 21,69 + 35,37 + 637,5 + 9350 = 22800,76 (руб.).

Таким образом, сумма расходов на разработку веб-приложения составила 22800,76 рублей.

## 6.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию веб-приложения Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса (формула 6.12).

|  |  |  |
| --- | --- | --- |
|  | . | (6.12) |

Срса = 22800,76 ⋅ 10 / 100 = 2280,076 (руб.).

где Срса – сумма расходов на сопровождение и адаптацию веб-приложения, руб.;

Ср – сумма расходов на разработку веб-приложения, руб.;

Нрса – норматив расходов на сопровождение и адаптацию, %.

Таким образом, сумма расходов на сопровождение и адаптацию веб-приложения составляют 2280,076 рублей.

## 6.3.11 Полная себестоимость

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию веб-приложения Срса (формула 6.13).

|  |  |  |
| --- | --- | --- |
|  | . | (6.13) |

где Сп – полная себестоимость веб-приложения, руб.;

Ср – сумма расходов на разработку веб-приложения, руб.;

Срса – сумма расходов на сопровождение и адаптацию веб-приложения, руб.

Сп = 22800,76 + 2280,076 = 25080,836 руб.

Полная себестоимость составила 25080,836 рублей.

## 6.3.12 Определение цены, оценка эффективности

При расчете прибыли будем опираться на среднюю рыночную цену, по которой можно разработать данное программное обеспечение. Средняя рыночная цена составляет 49000 руб.

Прибыль рассчитывается по формуле 6.14:

|  |  |  |
| --- | --- | --- |
|  | . | (6.14) |

где Ппс – прибыль от реализации программного средства, руб.;

Цр – средняя рыночная цена продукта, руб.;

Сп – полная себестоимость программного средства, руб.;

Ппс = (49000 / 1,2) - 25080,836 = 15752,497‬ (руб.).

Уровень рентабельности разработанного программного средства определяется по формуле 6.15:

|  |  |
| --- | --- |
| , | (6.15) |

где Урент – уровень рентабельности программного средства, %

Сп – полная себестоимость программного средства, руб.;

Ппс – прибыль от реализации программного средства, руб.

Урент = (15752,497 / 25080,836) ⋅ 100 = 62,8%

Рентабельность разработки программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе.

6.4 Вывод по разделу

В таблице 6.3 представлены результаты расчетов для основных показателей данной главы в краткой форме.

Таблица 6.3 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 5 |
| Количество программистов, чел. | 1 |
| Зарплата с отчислениями, руб. | 12794,28 |
| Расходы на материалы, оплату машинного времени, прочие, руб | 694,56 |
| Накладные расходы, руб | 9350 |
| Себестоимость разработки веб-приложения, руб | 22800,76 |
| Расходы на сопровождение и адаптацию, руб. | 2280,076 |
| Полная себестоимость, руб. | 25080,836 |
| Цена аналога, руб. | 49000 |
| Прибыль от реализации, руб. | 15752,497‬ |
| Рентабельность разработки, % | 62,8 |

Необходимость разработки данного программного обеспечения обусловлена тем, что при создании мобильного приложения прежде всего необходимо нарисовать прототип интерфейса. Для того чтобы этот прототим был максимально удобным и пользным при разработке, он должен быть интерактивным, и удобным в распространении. А так разработанное приложение позволет создавать интерактивные интерфейсы в виде мобильного приложения, это позволит увидеть вживую то, как будет выглядеть тот либо иной элемент на мобильном устройстве.

Разработка программного средства одним работником в течение пяти месяцев при заданных условиях обойдется в 25080,836 рублей. Исходя из полученных данных, можно сделать вывод, что программное средство является экономически выгодным и рентабельность в 62,8% говорит о достаточно высокой окупаемости вложенных средств в реализации данного программного средства.

**Заключение**

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Рыжанкова А.С.*

*Вольский Д.М.*

*Жиляк Н.А.*

Ф.И.О.

Подпись

Дата

1

*Заключение*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*74218005, 2020*

В рамках дипломного проекта был произведен патентный поиск. На рынке существуют решения для создания интерактивных интерфейсов мобильных приложения, но все они имеют те или иные недостатки. В итоге было разработано веб-приложение для генерации интерактивных интерфейсов под операционную систему Android, отвечающее всем параметрам, важным для удобной работы пользователей. В рамках веб-приложения пользователю доступны следующие возможности:

– создание интерактивных интерфейсов;

– управление проектами;

– генерация мобильного приложения.

В рамках данной дипломной работы решены следующие задачи:

* разработана структура интернет-ресурса;
* выбораны и обоснованы средства разработки;
* разработан пользовательский интерфейс;
* спроектированна база данных для работы пользователя с интернет-ресурсом с целью осуществления контроля над проектами и процессом их создания, редактирования и удаления;
* разработана генерация мобильного приложения на основе созданного проекта.

Представлено описание каждого этапа проектирования и реализации веб-приложения, которое позволит осуществлять генерацию интерактивных интерфейсов для операционной системы Android без каких-либо усилий.

В пояснительной записке к дипломному проекту приведено описание руководства пользователя разработанного веб-приложения, рассчитаны затраты на разработку представленного в дипломном проекте веб-приложения и прибыль от его реализации.

*Жиляк Н.А.*

Консульт.

# Список использованной литературы

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Рыжанкова А.С.*

*Вольский Д.М.*

*Жиляк Н.А.*

Ф.И.О.

Подпись

Дата

1

*Список использованной литературы*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*74218005, 2020*

1 Metanit [Электронный ресурс]. – Режим доступа: metanit.com/. – Дата доступа: 15.03.2020.

2 Обзор ASP.NET MVC 5 [Электронный ресурс]. – Режим доступа: habrahabr.ru/. – Дата доступа: 01.04.2020.

1. Начинаем изучать ASP.NET MVC [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/ASP\_NET/mvc/level1/. – Дата доступа: 04.03.2020.
2. Model-View-Controller [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Model-View-Controller/. – Дата доступа: 15.03.2020.
3. Многоуровневая архитектура в ASP.NET MVC [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/mvc5/23.5.php/. – Дата доступа: 20.04.2020.
4. Onion-архитектура в ASP.NET MVC 5 [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/mvc5/23.2.php/. – Дата доступа: 21.04.2020.
5. Энциклопедия языков программирования [Электронный ресурс]. – Режим доступа: http://progopedia.ru/language/sql/. – Дата доступа: 15.03.2020.
6. Введение в Entity Framework [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/entity-framework/6/level1/. – Дата доступа: 18.03.2020.
7. Современный учебник JavaScript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/. – Дата доступа: 15.03.2020.
8. JavaScript [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/JavaScript/. – Дата доступа: 15.03.2020.
9. JQuery [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/JQuery/. – Дата доступа: 14.03.2020.
10. JavaScript и jQuery – HTML5Book [Электронный ресурс]. – Режим доступа: https://html5book.ru/javascript-jquery/. – Дата доступа: 16.03.2020.
11. Bootstrap3 Tutorial [Электронный ресурс]. – Режим доступа: w3schools.com/. – Дата доступа: 15.04.2020.

Жиляк Н.А.

Консульт.

ПРИЛОЖЕНИЕ А

Блок-схема алгоритма изменения проекта

ПРИЛОЖЕНИЕ Б

Блок-схема алгоритма генерации мобильного приложения

ПРИЛОЖЕНИЕ В

Блок-схема алгоритма добавления элемента на страницу реактирования

ПРИЛОЖЕНИЕ Г

Блок-схема функции добавления нового проекта в базу данных

ПРИЛОЖЕНИЕ Д

Диаграмма последовательности работы приложения

ПРИЛОЖЕНИЕ Е

Исходный код генератора мобильного приложения

public byte[] Design(out string timeFolderToApk)

{

#region Copy template project and parse json' core

Status = BuildStatus.Started;

\_builder.WriteLogs($"{DateTimeOffset.Now}: Got buildSettings:\n {\_buildSettings.ToJSON()}");

Utils.CopyDirectory(\_buildSettings.TemplateProjectFolder, \_buildSettings.TempProjectFolder);

\_builder.WriteLogs($"{DateTimeOffset.Now}: CopyDirectory finished: \n" +

$"oldpath = {\_buildSettings.TemplateProjectFolder}\n" +

$"newpath = {\_buildSettings.TempProjectFolder}");

Status = BuildStatus.ParsingProjectJson;

var baseJson = \_json.TryFromJSON<BaseJson>();

if (baseJson.Permissions == null)

baseJson.Permissions = new List<string>();

if (baseJson.Credentials == null)

baseJson.Credentials = new Dictionary<CredentialType, string>();

//baseJson.Models?.ForEach(m => m.Name = Utils.ConvertNameToValidClassName(m.Name + $"\_{Guid.NewGuid()}"));

\_builder.WriteLogs($"Recieved {baseJson.Pages.Count()} pages, start parsing.");

#endregion

var packages = new List<PackageInfo>();

var customFiles = new List<CustomFileInfo>();

var manifestPerm = new List<StoredValue>();

var manifestBody = new List<StoredValue>();

string onCreate = string.Empty;

#region First pages' parsing & PagesManager's init

var pages = new List<Page>();

var solutionManager = new SolutionManager(\_buildSettings.SolutionName, \_buildSettings.PathToCoreFolder, baseJson.Permissions);

var langs = baseJson.Languages ?? new List<string>();

langs = langs.Count == 0 ? new List<string>() { "Unnamed language" } : langs;

solutionManager.Texts.SetLanguages(langs);

solutionManager.Models.SetModels(baseJson.Models?.ToArray() ?? new ModelJson[0]);

ModelHelper.GetAllModelsAvailable();

// First time we should just get full list of pages with minimum page settings

foreach (var jTokenPage in baseJson.Pages)

{

try

{

var page = Page.Create(\_buildSettings, jTokenPage, solutionManager);

pages.Add(page);

\_builder.WriteLogs($"Page '{page?.Name}' parsed sucessfully. Maybe. If not null.");

}

catch (Exception e)

{

\_builder.WriteLogs($"Exception:::{Environment.NewLine}{e}{Environment.NewLine}{jTokenPage.ToJSON()}");

}

}

// Add all the pages to special manager to set navigation and other

solutionManager.XFPages.SetPages(pages);

// Now set App Icon and App name.

var initA = new Android.DefaultFilesInitiator(baseJson.AppName,

\_buildSettings.SolutionName,

baseJson.AppIconBase64,

\_buildSettings.AndroidProjectFolder)

{

SplashScreenImageBase64 = baseJson.SplashScreenImageBase64,

SplashScreenImageMargin = baseJson.SplashScreenImageMargin,

SplashScreenBackgroundColor = baseJson.SplashScreenBackgroundColor

};

initA.Process();

initA.WriteAllToFiles(\_buildSettings);

var initC = new Core.DefaultFilesInitiator(baseJson.AppName,

\_buildSettings.SolutionName,

\_buildSettings.PathToCoreFolder);

initC.Process();

initC.WriteAllToFiles(\_buildSettings);

var initI = new Ios.DefaultFilesInitiator(baseJson.AppName,

\_buildSettings.SolutionName,

\_buildSettings.IosProjectFolder);

initI.Process();

initI.WriteAllToFiles(\_buildSettings);

#endregion

#region Process each page and fulfilling them with children controls

// Second time we process each page and fullfilling it with controls and content

foreach (var page in pages)

{

page.AddControlsFromJTokens(page.JChildren);

page.TryWriteToCoreFolder(\_buildSettings.PathToCoreFolder, out string logs);

\_builder.WriteLogs(logs);

customFiles.AddRange(page.WriteAllCustomFilesInside());

packages.AddRange(page.GetPackages());

manifestBody.AddRange(page.GetAndroidManifestApplicationBody());

manifestPerm.AddRange(page.GetAndroidManifestPermissions());

onCreate = string.Concat(onCreate, page.GetAndroidMainActivityOnCreate());

}

#endregion

#region Create App page

// Now we need set start page of the application

var startPageName = solutionManager.XFPages.GetPageNameById(baseJson.MainPageId);

if (string.IsNullOrWhiteSpace(startPageName))

{

// TODO: Decide, maybe better throw exception?

// For example throw in DEBUG and take random page in Release.

// Or just always throw new ArgumentException.

\_builder.WriteLogs("WARNING::: Start page was not set, try to set random page as start page.");

startPageName = pages.FirstOrDefault(p => !p.IsPopup)?.Name;

if (string.IsNullOrWhiteSpace(startPageName))

{

throw new ArgumentException("Some problems with parsing pages: pages (not popups) count = 0!");

}

}

var appPage = new AppPage(solutionManager, startPageName);

appPage.TryWriteToCoreFolder(\_buildSettings.PathToCoreFolder, out string log);

\_builder.WriteLogs(log);

appPage.WriteAllToFiles(\_buildSettings);

#endregion

#region Setting up other part of solution

var customD = new Android.PostFilesInitiator(baseJson.AppName,

\_buildSettings.SolutionName,

\_buildSettings.AndroidProjectFolder, packages, customFiles, manifestPerm, manifestBody, onCreate);

customD.Process();

customD.WriteAllToFiles(\_buildSettings);

var customC = new Core.PostFilesInitiator(solutionManager, baseJson.AppName,

\_buildSettings.PathToCoreFolder, packages);

customC.Process();

customC.WriteAllToFiles(\_buildSettings);

// Now add Translate resources manager class

var textResourcesGenerator = new TextResourcesGenerator(\_buildSettings.SolutionName,

solutionManager.Texts.Languages,

solutionManager.Texts.Resources);

textResourcesGenerator.Process(\_buildSettings.PathToCoreFolder);

#endregion

#if DEBUG

var ttt = \_buildSettings.TempProjectFolder;

#endif

#region Try build apk

try

{

Status = BuildStatus.Building;

var pathToApk = \_builder.Build();

Status = BuildStatus.CompletedSuccessfully;

timeFolderToApk = new FileInfo(pathToApk).Directory.Name;

//var link = $"http{(Request.IsHttps ? "s" : "")}://{HttpContext.Request.Host.Value}/Api/Build/Download/{timeFolderToApk}";

//builder.WriteLogs("timeFolderToApk|" + link);

return File.ReadAllBytes(pathToApk);

}

catch (Exception ex)

{

Utils.WriteErrorToConsole(ex);

Status = BuildStatus.Failed;

timeFolderToApk = null;

return new byte[0];

}

finally

{

CurrentLogs = \_builder.GetCurrentLogs();

}

#endregion

}